Navigating Networks using Overlays

Colin Stephen Myers

Submitted in accordance with the requirements
for the degree of
Doctor of Philosophy

The University of Leeds
School of Computing

February, 2013

The candidate confirms that the work submitted is his own, except where work which has formed part of jointly-authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Parts of the work presented in this thesis have been published in the article below. The content of the article is the candidate's own work which was written under the supervision of Dr. David Duke. Chapters 1 and 3 make reference to some of the ideas described in the article, and the example data forms the basis of a case study in Chapter 5.

Colin Myers and David Duke, "A Map of the Heap: Revealing Design Abstractions in Runtime Structures." In *Proceedings 5th ACM International Symposium on Software Visualization*, pages 63–72. 2010.

Acknowledgements

This thesis is dedicated my grandparents Harry and Edna Ducker, whose thoughtfulness, compassion, and stoicism have been a constant inspiration. I must also give special mention to my sister Charmaine who provided much needed encouragement and understanding during my studentship, along with occasional financial assistance.

My deepest gratitude goes of course to my academic supervisor Dr. David Duke, whose combination of high expectations and patience have been essential components in the long journey that led to the production of this thesis. My development as a researcher has benefited greatly from his experience and thoughtful insights. I would also like to thank Emeritus Professor Ken Brodlie who first sparked my interest in information visualization during his lectures. Being a member of the Visualization and Virtual Reality research group has been a great privilege and the many conversations, presentations, and tea and biscuits shared with staff and students too numerate to list here is much appreciated, and has surely helped to sharpen my thinking.

Abstract

This thesis contributes a novel approach to navigation tasks in large graphs. Graph visualization is the problem of representing the structure of a mathematical graph $G = (V, E)$, V a set of vertices (or nodes) and $E \subseteq V \times V$ a set of edges. My work is concerned with the node-link representation of graphs and I use the term *network* to distinguish this external representation from the underlying mathematical structure. Networks are an intuitive representation of a set of elements and the relationships between them, and are known to be effective for analysis tasks involving following paths between nodes. I define *navigation* as the task of identifying and following such a path in display space.

Unfortunately the utility of a network diminishes as the density of edges increases and edge-crossings make navigation taxing. A well-explored approach to this problem is to find a perspicuous layout of the nodes. While this improves the readability of individual nodes and edges it may also require a compromise: to be easily understood the overall arrangement of the network should also correspond with the user's internal mental model of the domain, a property referred to as *congruence*. Other solutions distort the display space or use multiple-scaled-views to promote comprehension of local details while retaining awareness of the global context, but often lack direct support for navigation of the network topology beyond the local context.

This thesis contributes a model of visual graph analysis that brings together recent advances in cartographic representation, diagram comprehension, and graph visualization, leading to a greater understanding of network navigation bottlenecks in terms of the degree of correspondence between the external graph representation, and the user's 'mental map'. Motivated by this model I present a new approach to graph visualization that separates concerns of navigation from those of depiction with the aim of improving correspondence between the internal and external representations. I describe the design and realization of an interface for network navigation inspired by the new approach within a pipeline-based architecture, and provide a reflective evaluation of the implementation.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

I begin this chapter by introducing the purpose of visualization and outline the principle issues of data, representation, and implementation. This leads to a discussion of the general problems of visual scalability and its impact on the main focus of this thesis: *navigation*. Graph visualization is a challenging, well-studied research topic and several graph representations and navigation tools are available, which are outlined here. The chapter concludes with an introduction to my approach to the network navigation problem, and a statement of the scope and contributions of this thesis.

1.1 Visualization

Visualization refers to the process of using an external visual representation to enhance cognition by externalizing memory, and by arranging graphical marks in space such that visual cues enable useful interpretations of the data [21]. Bergeron [19] sets out three uses of visualization:

descriptive visualization is used to present evidence to others of some phenomenon known to exist in the data set;

analytical visualization is the process of searching the data for evidence that either confirms or refutes a known hypothesis; and,

exploratory visualization is used when the user has no particular hypothesis about the data and wishes to understand what is present by, for example, recognizing items,

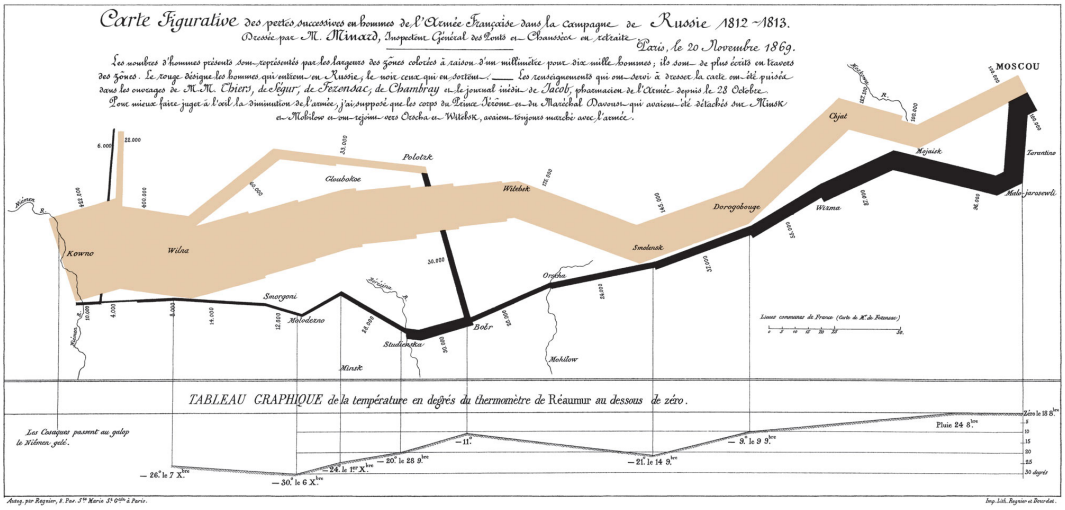

Figure 1.1: Minard's figurative map from 1869 combines depictions of time, geography, temperature, and mortality to tell the story of Napoleon's failed campaign on Russia. (Figure licensed under Creative Commons).

> or identifying patterns, outliers and trends in the data, so an hypothesis may be formed.

The capacity of modern visualization systems to support these tasks emerges from the interaction between a computer, which provides an interface to rapidly refine and encode data as graphical marks on the screen, and the human user, who has an innate ability to perceive and infer meaning from those marks.

Hand-crafted visualization artefacts have been used for data presentation for at least two centuries: Minard's 1869 map for example presents data in several dimensions to illustrate the fate of Napoleon's campaign on Russia. In recent decades visual data presentation has become a topic of formal study, leading to the formulation of data presentation heuristics based on Gestalt theories of spatial grouping, and works such as Tufte's "Theory of Data Graphics" [125] and Bertin's classification of "retinal variables" [14]. An early approach to computerized visualization sought to automate the process of visualization design. Rules of visual encoding and diagram construction were encoded and used by the system to calculate and display the most "effective and efficient" representation given a particular input data-type [79], with later models also incorporating rules based on the user's task [22].

Limited by graphical processing power, early systems concentrated on presenting a static view of the data. Tukey [126] had already demonstrated the utility of *exploratory data analysis*, using data plots to discover an hypothesis in contrast to the earlier practice

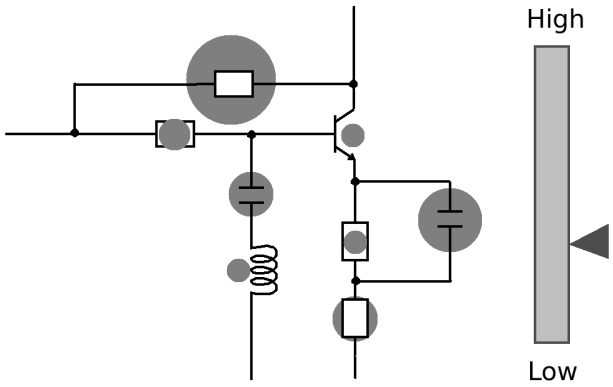

Figure 1.2: An example of exploratory visual analysis of a complex system. In this simulated circuit diagram the grey circles change size to represent the performance of each component given a particular input signal frequency. The input frequency is adjusted using the slider control on the right of the diagram. In this way the user can continuously explore the range of input values, observing the effect of changes in input instantaneously. (Figure adapted from [117]).

of merely presenting findings to others. Meanwhile Bertin [14] described a process of *visual thinking*, constructing a matrix of data values and permuting the columns until patterns emerge that could indicate trends or reveal outliers. Once an interesting permutation is found it is the job of the analyst to reason about what information the patterns reveal, in the context of their domain knowledge.

Bertin initially conducted visual exploration using specially prepared cards on a tabletop, only moving to computerized support after more than a decade. Exploratory analysis is where computerized visualization provides the greatest benefit, enabling visual thinking through interactive tools that allow the user to filter, group and rearrange the display on demand. Spence's electronic circuit diagram [117] is a compelling example of interactive visualization that enables insight into a complex phenomenon by changing the visual display in response to user inputs (see Figure 1.2).

1.1.1 Scientific Visualization

The formal study of visualization in computer science is generally attributed to the publication of a synopsis of a National Science Foundation advisory report on "Visualization in Scientific Computing" [83]. The report identified that scientists were producing increasing volumes of output from computational simulations yet few tools were available to assist them in analysing large, complex sets of data. Visualization offered the possi-

bility of transforming the raw symbolic data into geometric forms that are more readily interpreted by the scientist. Furthermore, the ability to steer computation during processing was required so that the effects of parameter changes were immediately visible, to not only *see* the data, but also to *interact* with it in real-time.

Scientific visualization as it has come to be known, is used to analyse numerical data such as the output of a medical MRI scan, or an engineering simulation. It may be categorized as being concerned with models of *continuous* data with the consequence that sample points may be interpolated in a meaningful way [124].

1.1.2 Information Visualization

The term "Information Visualization" was first coined by Robertson, Card and MacKinlay [104] to mean, "the direct manipulation of information objects and the structure between them.". While scientific visualization was aimed at analysis of the continuous models produced by scientists, increasing attention was being given to visualizing data from discrete structures such as the tabular data and document collections found in commercial databases. An early example of this is SemNet [44], a graph visualization tool aimed at exploring the relationships between items in a large knowledge base.

The data represented by information visualization systems represent a space that is *abstract* [42], and therefore has no direct mapping to the geometric structures used in graphics. The information visualization designer must therefore select an appropriate spatial metaphor. Popular spatial metaphors include landscapes [8], cities [120], and trees. The use of metaphor need not be consistent or realistic: consider the now familiar metaphor used in the graphical interface of a personal computer: groups of "files" are contained in "folders" which are placed on the surface of a "desktop". To interact with a file or folder, its contents are displayed in a "window". The mixed metaphors (folders arranged on a desk versus looking through a window) do not seem to be a barrier to effectiveness, as each one is appropriate to a particular activity. Figure 1.3 contains an example of visualizations of the same underlying data using both a continuous model to form a 3D volume rendering and a discrete model using a tree metaphor.

A key challenge in information visualization is the provision of tools to *navigate* the space the diagram is drawn in. Navigation refers to the "process of selecting and following a path in display space" [70]. While the drawing canvas is an infinite plane the computer display is constrained by its dimensions and pixel resolution. This often means

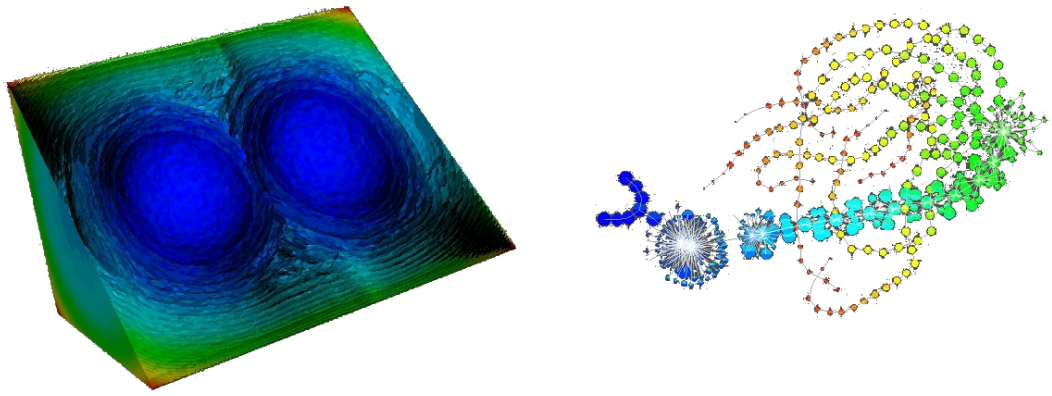

Figure 1.3: Fission of a plutonium nucleus represented using a continuous model and rendered as a 3D volume (left), and a discrete model with a tree metaphor (right). The discrete model indicates critical points in the vector field which can then be related to the spatial position of phenomena as displayed in the volume.

that the whole data cannot be displayed in sufficient detail on a single display, as doing so would present the viewer with an unintelligible mass of graphical marks. To combat this, pan and zoom tools are commonly employed so the user may choose what portion of the diagram is visible in the display, but this can lead to "desert fog", the condition that there are insufficient visual marks to enable the user to navigate successfully [70]. With no recognizable visual references the user becomes lost in the visualization and may be unable to answer navigation questions such as, "where am I now?", "where do I go next?", or "how do I get back to where I started?"

1.2 Graphs and Networks

Graph visualization refers to the problem of representing a mathematical graph $G = (V, E)$, V a set of vertices and $E \subseteq V \times V$ a set of edges. A graph is a formal mathematical model of data items and relationships between those data items. The surveys by Herman et al [64] and Landesberger et al [134] described a number of graph representations, which of these to use to depends on structural properties of the graph and on the task at hand.

- Containment diagrams such as tree-maps [111] allow a compact representation of hierarchical data.
- Adjacency matrices reveal patterns of different shaped blocks (clusters of incident edges) provided an appropriate ordering of the rows and columns can be

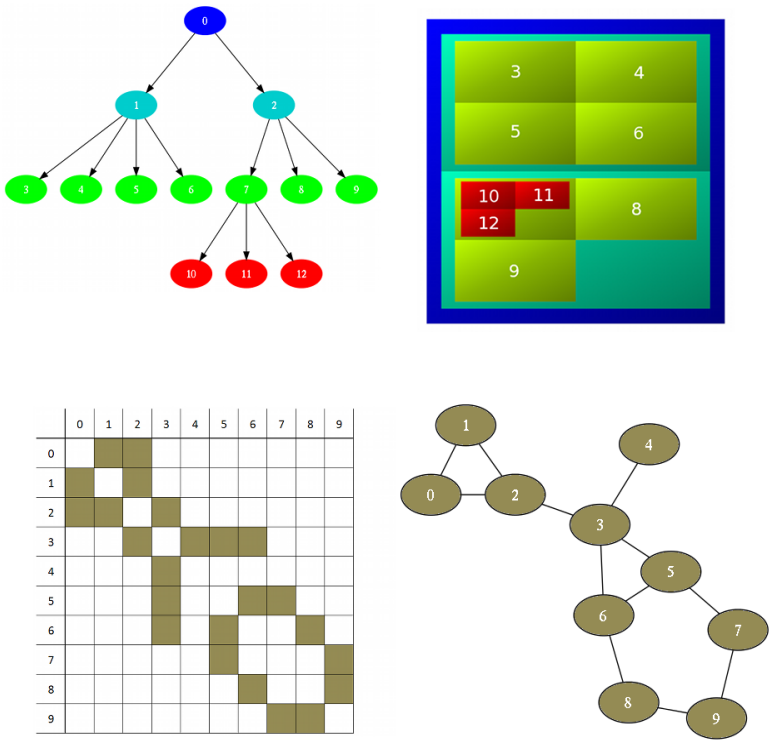

Figure 1.4: (Top) Comparison of network and containment representations of the same hierarchical data. (Bottom) Comparison of matrix and network representation of the same graph data. In both cases, the network representation is more effective when following paths between items.

found [62].

- Node-link diagrams, where vertices are represented by glyphs, and edges by polylines or splines, are appropriate for tasks related to paths formed by sequences of adjacent vertices [54].

Figure 1.4 shows a comparison of different representations of the same graph and tree data.

This thesis deals exclusively with the node-link representation of graphs, and following Bertin [14], I use the term *network* to distinguish this external representation from the underlying mathematical structure. Networks provide an intuitive spatial metaphor of a discrete set of related elements by taking advantage of basic human visual principles of enclosure to represent a contained object and connectedness to indicate a relationship between them [129].

1.2.1 Layout

A layout algorithm assigns coordinates to each element of a graph with the aim of producing an effective and efficient network drawing. *Aesthetics* are drawing rules encoded in the algorithm that constrain the placement of graphics primitives [33]. Computing the coordinates of a layout to optimize aesthetics is well-studied (see [33] for a comprehensive survey and [134] for more recent advances).

A primary aesthetic concern is to produce a network in which edges can be easily followed between nodes. User studies by Purchase [98] and others have shown that the number of bends in edges and the number of edge-crossings significantly reduce the accuracy of following paths in a network. Beyond minimizing edge-crossings, there are too few empirical studies of aesthetics to underpin a set of general layout principles.

1.2.1.1 Node Layout

Since the general optimization problem is NP-hard [64], various approximation approaches to node layout have been devised. Sugiyama et al [122] introduced a layout method for directed acyclic graphs that first positions nodes in horizontal layers so that all edges point downwards. A second pass re-orders the nodes to minimize the number of edge crossings between layers. Force-directed methods were introduced by Eades [39] with improvements by Kamada and Kawai [71], and Fruchterman and Reingold [48] amongst others. These algorithms model the network as a mechanical system where edges are treated as springs that repel vertices that are close together and pull together vertices that are distant. A minimal total energy of the system is then computed iteratively. The running time complexity of these methods is quadratic and therefore prohibitive on large graphs. To overcome this problem modern force-directed algorithms use the fast multi-pole multilevel method (FM^3) introduced by Hachul and Jünger [58] that lays out a general graph in $\mathcal{O}(|V|log|V|+|E|)$. The speed-up is achieved by first solving a coarsened representation of the original graph and then iteratively refining the solution until the original input graph is laid out.

1.2.1.2 Edge Layout

The simplest edge representation draws a straight line between the two end-points of an edge. Parallel edges may be drawn as parallel arcs to avoid over-plotting. Edge-

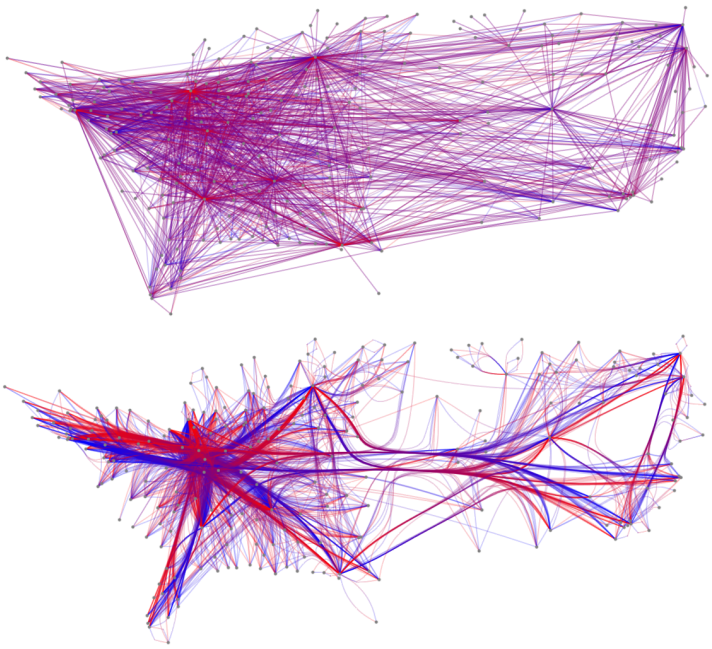

Figure 1.5: A graph of US airline routes rendered with conventional straight-line edges (above) and bundled edges (below). The bundles reduce the cluttering effect of edge crossings by closely grouping edges that have a similar start and end point. Following individual edges requires additional tools to separate them. The method shown here [110] separates the bundles according to the direction of edges (direction is encoded as a blue-to-red colour ramp).

routing algorithms introduce splines so that edges can be routed around obstacles [35]. Recently, several techniques similar to “hierarchical edge bundles” [65] have emerged that reduce overall clutter by very closely grouping edges that have similar start and end points. For some specific applications highly specialized representations have been developed. In Agrawala and Stolte’s route-map system [2] edges (representing roads) are carefully distorted to shorten them and simplify their shape to ensure important junctions are clearly readable. For cases where edge attributes are of primary concern to the user, an alternative design is to duplicate nodes and vertically list all the edges in the centre of the display [14, 97].

1.2.1.3 Semantics

In addition to readability of the edges of the network it is important to note that the overall layout affects what meaning is suggested by the drawing. Topological features

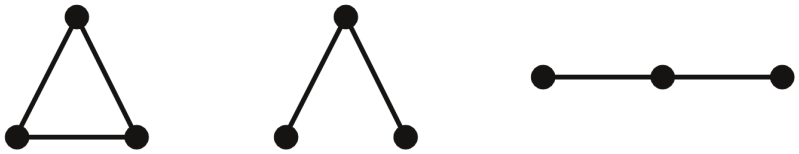

Figure 1.6: The relative positions of nodes suggests meaning. From left to right: a cycle implies equal status; the top node is superior to the two lower nodes; and, an ordered sequence. (Adapted from [32]).

such as symmetry or the presence of highly-connected subgraphs can be revealed with an appropriate layout. Moreover, certain spatial arrangements invoke particular meaning based on Gestalt principles and well-known conventions. Upon initial viewing, before the fine details of the edges are perceived, nodes that are in close proximity will be perceived as a group. In a circular layout all nodes have equal status, a linear arrangement implies an ordered sequence, and a top-bottom arrangement suggests a hierarchical relationship [32] (see Figure 1.6). Therefore, to be most effective a network must be readable and also should be arranged so that unintended inferences are avoided. Put another way, a network should be structured in a way that is consistent with the user's internal mental model of the domain it represents: a property referred to as *congruence* [78, 128].

To satisfy the congruence property graph layout algorithms that depend solely on topological structure require that the topology is in some way analogous to the intended meaning. In the case of a hierarchy such as a tree, the order of nodes in the drawing should mirror the rank of nodes in the hierarchy. In many cases however, the semantics of the graph are not made explicit by its topology and so the effects of layout may be misleading. Arguably, the most effective drawings of graphs are hand-curated as they may be refined over many iterations and incorporate domain conventions, see for example the KEGG pathway maps [72]. However, hand-crafting a non-trivial network is time-consuming, requiring many hours of refinement.

More specialized automated graph layout may be achieved by making some assumptions on the connection between topology and meaning. Schreiber et al [108] add constraints between substructures in the graph that reflect well established domain drawing conventions. Bespoke visualization systems can be developed to support highly specialized domain requirements (e.g. [90, 96]) by specifically encoding knowledge elicited from domain experts and designing an interface that supports a set of well-defined tasks.

1.2.1.4 Non-Graph Features

Additional semantic content can be given to networks by displaying non-graph features. Commonly, text labels positioned close to nodes or edges are used to display nominal or quantitative values associated with them. For some specialist applications such as class diagrams in software engineering, nodes are drawn as a significant geometrical shape, with details shown as text contained within the node [47]. Quantitative values attached to graph elements can also be rendered as "retinal variables" [14], varying edge-thickness or the area of nodes. Perceptual grouping other than spatial clusters can be supported by varying node shape or colour to differentiate classes.

Spatially grouped clusters of nodes can be collapsed and represented as single meta-nodes [40]. An alternative to collapsing nodes is to delineate clusters by adding regions of colour or texture to the substrate of the network to give a map-like appearance [51]. There are cases where the nodes represent some naturally spatial class of items, such as the airline map in Figure 1.5. Since the layout of the nodes is related to their geographical position, this type of network can be embedded in a geographical map. The addition of non-graph features represents a continuum of visual complexity from a plain, node-link diagram, through those imbued with retinal variables, to embedding in a continuous substrate.

1.2.2 Navigation in Networks

Navigating a large network is a frequent and non-trivial task. Lee et al's "Task Taxonomy for Graph Visualization" [74] summarizes commonly performed graph tasks and shows how they are composed of common low-level tasks. Many of the low-levels tasks they describe involve *finding* a particular node (perhaps using a search facility) and then *following* an edge to some adjacent target node. The tasks are cast in terms of topological navigation but are conducted in the display space, with the consequence that the ease with which one can navigate a network is related to the effectiveness of the results of layout. This problem becomes particularly relevant in the case that the network is too large to fit within the display at a readable level of detail. If we consider the network display and pan and zoom tools as an interface into the graph dataset, the problem becomes an instance of Norman's "Gulf of Execution" [92]: the vocabulary of the interface (pan, zoom, etc.) is semantically distant from the intention of the user (find this node, follow that edge).

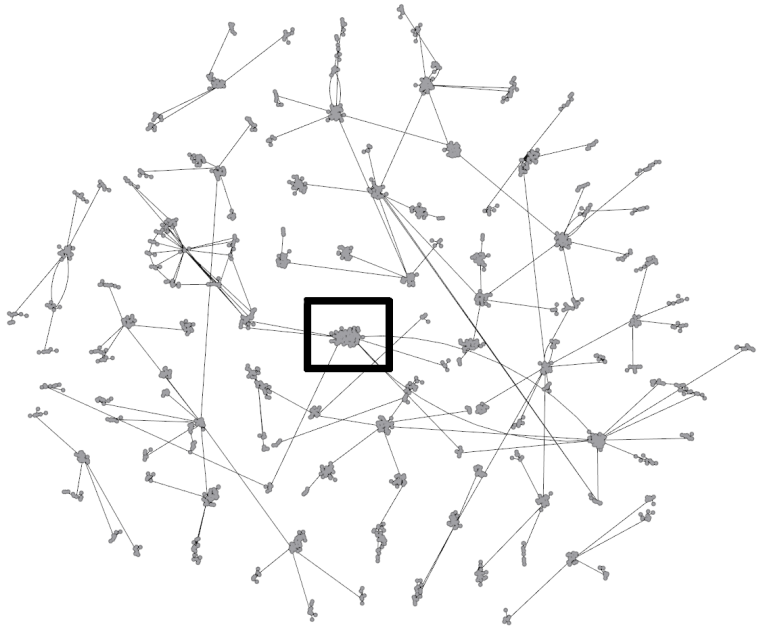

Figure 1.7: Overview of a network. While relations between spatial groups of nodes can be perceived, local connectivity between individual nodes is not readable. The group outlined by the black rectangle is enlarged in Figure 1.8.

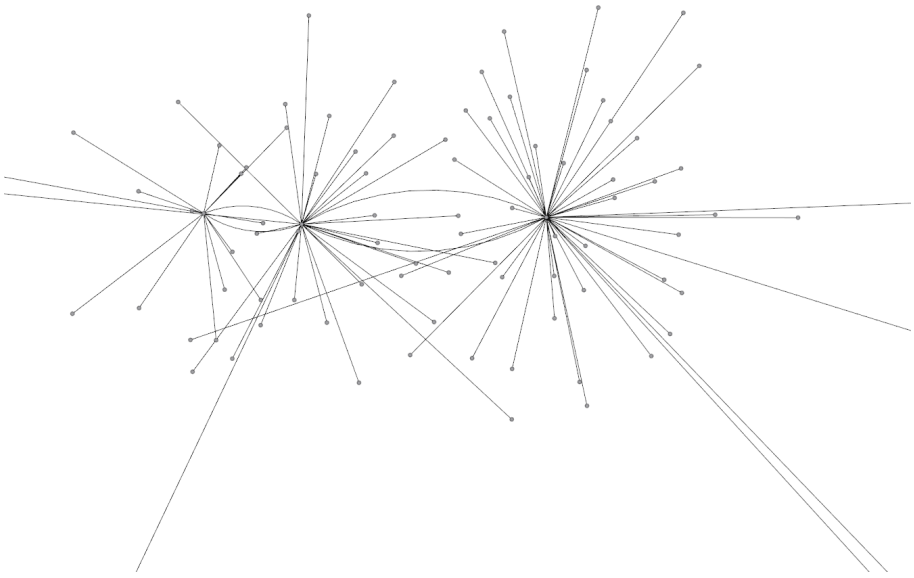

Figure 1.8: Detail view of the group outlined in Figure 1.7. At high zoom individual nodes and edges are easily followed but their relationship with the wider context of the network is lost.

Consider a scenario in which a user wishes to compare and contrast two similar looking subgraphs within a large network. A fully zoomed out view of the network may be sufficient to identify the *location* of the subgraphs in the overall space but to read the fine details of edges and labels she must zoom in to the first subgraph (see Figures 1.7 and 1.8). This means that the overall context is no longer visible. To then move to the second subgraph our user must recall its location in the overall network relative to the current position. In the absence of domain conventions and thus lacking congruence with the users internal spatial model, the user does not have a usable 'mental map' of the space occupied by the network. Instead she must suspend the primary analysis task and zoom out to regain the overall context so that the second subgraph can be located. Only after zooming in again on the second subgraph can analysis resume. The need to repeatedly switch task context disrupts the primary analysis task, and the user is forced to direct cognitive effort [73] towards navigation.

1.2.3 Solution Space

Given the difficulties encountered when required to navigate networks various strategies to aid the user have emerged.

Reduce the amount of graphical marks by *filtering* out unneeded data [113] or by instead displaying a suitable *abstraction*. A common abstraction method involves an hierarchical clustering of connected subgraphs into single meta-nodes [40], so that the connectivity between meta-nodes is depicted instead of the low-level details of the network.

Scale parts of the network to provide both *focus and context* [107] in the same view or provide multiple *overview and detail* views at various scales.

Enhance the visual appearance of the nodes and edges of the network by judicious use of colour, edge-routing, or rendering techniques.

Each of these methods changes the final appearance of the network by modifying the data model, the coordinate system, or the graphical encoding respectively. A complementary approach is to add further specific support for network navigation. For example,

Topological navigation adds specific interactive support for following edges between adjacent nodes [85].

Explicit landmarks added to the display highlight familiar semantic components within a network [91].

The techniques described in this section are discussed further in Chapter 2. This thesis is concerned with the latter solution: adding explicit landmarks to the display with the aim of improving the correspondence between the network display and the user's spatial mental model.

1.3 Thesis and Contributions

This thesis proposes a novel approach to the problem of network navigation implemented as a distinct layer of customized navigation support that promotes congruence by strengthening the link between the meaning encoded in the graph, the network display, and the user's spatial mental model. The design is grounded in theory of graph comprehension and uses ideas adapted from navigation in the real-world. I provide arguments and evidence that adding extra graphical marks to a graph visualization enhances navigability without compromising the primary network display with additional clutter.

Specifically, this thesis makes the following contributions:

- A model of visual graph analysis that brings together recent advances in cartographic representation, diagram comprehension, and graph visualization.
- Motivated by this model I present the design of a new approach to graph visualization as a separate navigation layer.
- Realization of the design in a pipeline-based architecture designed for general, large-scale visualization.
- A reflective analysis of the new visualization designs.

A Note on Software Implementation To support the development of this thesis a prototype network visualization system, known as "CoronaScope", was implemented that provides navigation in the form of a pan and zoom tool. The navigation overlay designs set out in Chapters 3 and 4 were implemented within the CoronaScope application. The software was developed within a demand-driven pipeline architecture [57, 109] that has traditionally been used for scientific visualization and only recently adapted to include components to support information visualization [139]. Since the use of this paradigm in information visualization is experimental, I report on my experiences in Chapter 6.

1.4 Thesis Organization

In this chapter I highlighted the benefits of visualization, introduced the problem of navigation in networks, and presented the background, aims and contributions of this thesis. In the remainder of this thesis each chapter begins with a brief synopsis of the topics covered, and concludes with a summary in which key issues and contributions are highlighted. A thorough overview may be obtained by reading the introductions, figure captions, and summaries in the order they appear. Briefly, the subject of this thesis is organized into the following chapters:

Chapter 2 contains an analysis of work related to this thesis, including models of information visualization, cognitive theory of how networks are comprehended, and existing solutions to the problem of network navigation.

Chapter 3 sets out the theoretical motivation for using landmarks to support network navigation, and heuristics for selecting landmarks from the structural and semantic content of the underlying network are proposed. The design of an overlay of visual landmarks is described and justified in terms of the cognitive basis.

Chapter 4 is concerned with the use of landmarks in a multi-scale visualization, that is, using a basic pan and zoom tool. I show how landmarks can be used to promote awareness of the global network structure using off-screen visualization, and go on to describe specific support for following network paths, and revisiting.

Chapter 5 looks at three case studies to demonstrate the proposed landmark and off-screen visualization tools, and contains an informal analysis of their efficacy.

Chapter 6 contains a detailed description of the overall software implementation, and

reports on my experience of using the pipeline paradigm to add navigation support to an existing information visualization technique.

Chapter 7 is a discussion of the contributions of this thesis and suggests future directions for the work.

Chapter 2

Related Work

This chapter begins with a review of some of the organizing principles of information visualization, and I highlight recent developments that incorporate theories from the related field of cognitive science. Inspired by MacEachren's multi-level approach to forming a comprehensive theory of map interpretation, I discuss results related to how networks are perceived and integrated with high-level inference processes. After outlining the some of the difficulties that arise in network navigation, the existing solution space is considered, along with specific design issues related to the use of overlays and transparency in visualization design. Finally, options for software implementation are reviewed.

2.1 Foundations of Information Visualization

2.1.1 Reference Models

When studying or describing a complex domain like information visualization, a reference model can help to identify the primary concepts of study, and provides a set of common principles and language for programmers, visualization designers and users alike [19]. The "Information Visualization Reference Model" [21] and Chi's contemporaneous "data state reference model" [24] both describe a model which begins with some set of data that is transformed between each stage in the model (see Figure 2.1). After filtering and structuring the data, it is mapped to geometrical objects that are arranged on a substrate. Finally, a view of the substrate is presented to the user. The parameters

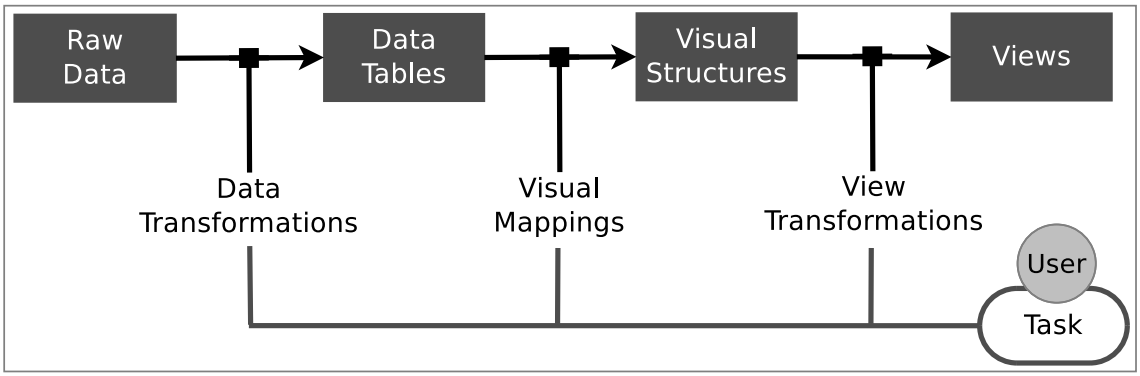

Figure 2.1: The information visualization reference model [21]. A user conducts a visualization task by interactively modifying the parameters of each of the transformations between data states.
Data transformations Raw data is processed and organized into relations and meta-data (graph and node/edge attributes).
Visual mappings Values are mapped to visual structures and positioned on a substrate (network).
View transformations The substrate is positioned, clipped and scaled to produce the final rendered image (view). (Figure adapted from [21]).

to each transform are interactively modified by a user according to some specific task, allowing them to choose a new sub-set of data or change their viewpoint. The similarity between the two reference models is significant, suggesting a strong consensus.

A taxonomy imposes structure on a domain by organizing the relevant methods into categories (a comprehensive survey of information visualization taxonomies can be found in [140]). Chi used his data states model to structure a taxonomy of information visualization techniques [23]. Additionally, Shneiderman [112] classifies data into several types and gives a list of tasks that encapsulate the high-level goals of users. On the basis that information seeking requires selection of items that satisfy a range of values Shneiderman summarizes the tasks in his mantra, “Overview first, zoom and filter, details on demand.” Ward and Yang [135] similarly define screen, data, and visual attribute spaces and identify interaction operators such as ‘navigate’, or ‘select’. Rather than focussing on presentation and interaction techniques others [140, 142] organize the topic from the point of view of the user’s analytic intent, giving rise to categories such as ‘cluster’, ‘compare’, and ‘rank’.

2.1.2 Distributed Cognition

While taxonomies and reference models are useful in generating or thinking about visualization design, a fully developed theoretical model is lacking in information visualization [75]. One review of the literature suggested that visualization designers have relied on a mélange of design guidelines [30] such as Shneiderman's mantra [112], Bertin's semiotics of graphics [14] and Tufte's graphic design rules [125]. Often the theories have focussed on the design and implementation of systems that respond to the user through interaction. *Cognitive science* is the study of the structure and processes of the human mind drawing together numerous disciplines including behavioural psychology, neuro-imaging and computational modelling. A deeper understanding of cognition as it relates to information visualization allows researchers to take a top-down perspective: how does the user respond to the visualization system? More recent work on the properties of the human visual system and how these affect the perception of visual displays show promising steps in this direction [101, 136].

The bottom-up and top-down perspectives are clearly not unconnected, just as interaction changes the visual display, so the user's internal state must change in response. Liu et al proposed *distributed cognition* as a theoretical basis for information visualization [75]. Unlike the traditional cognitive science view that cognition is internal while external spaces are merely input that must be encoded, distributed cognition extends the boundaries of study to include external stimuli as part of the range of cognitive resources available to the subject, allowing researchers to consider new ways in which people coordinate internal and external spaces. Liu and Stasko [76] also gave a definition of a mental model for information visualization that encompasses the visual and interactive properties of the external system as well as internal representations that link schematic, semantic and item-level information about the data. They also proposed that the study of interaction in information visualization should include three primary processes:

external anchoring is the act of locating a feature in the external image upon which we superimpose or *project* an internal image, such as extrapolating an imaged line (project) from a plot-line to intersect the axis mark (anchor);

information foraging involves reconfiguring and exploring a visualization; and,

cognitive off-loading means that internal working memory is transferred to the external representation.

In his seminal book on reading and understanding maps, MacEachren [78] draws to-

gether several theories and empirical results at multiple levels of abstraction, from low-level perception via cognitive theories of visual reasoning, to theory of knowledge representation. MacEachren links this top-down theory with the more traditional cartographic approach of semiotics (*this* means *that*), to show how the symbols and other visual details in maps are associated with specific meaning. Inspired by MacEachren's systematic approach, the following section considers some of the existing evidence on *how networks work*.

2.2 Reading and Understanding Networks

2.2.1 How Networks are Seen

Ware's book [136] provides a thorough overview of theories and models of perception that relate to information visualization and describes how general theories of perception such as Gestalt grouping [43] can inform visualization design. Individual perceptual theories have been the subject of user studies specifically for information visualization, for example Healey et al's work [60] on pre-attentively (automatically and in parallel) estimating the number of items in categories, where categories are delineated by hue or orientation. Others showed that spatial arrangement is a limiting factor in the effectiveness of pre-attentive abilities [59]. Beyond pre-attention, often referred to as pop-out, there is no consensus on the salience of features and which are more likely to be encoded. May et al [82] propose a hierarchical decomposition with significant top-level objects being composed of parts. A deliberate shift in attention requires that the viewer must first move to a top-level object before accessing any of its consituent parts.

Purchase [98] compared different graph drawing aesthetics to see which was the most "important to human understanding" by conducting controlled experiments measuring time and error on basic topological tasks such as following a path between two nodes or identifying cut-edges. Ware et al [138] found that while short paths of up to a few nodes can be read in parallel (at a glance), bends, edge-crossing, and number of branches all add to the cognitive cost for longer paths. They conclude that these features can be subjected to a trade-off: estimating the cognitive cost of a 38° bend is equivalent to one edge crossing.

Eye-tracking studies have been conducted to understand the gaze patterns employed by users when following paths in a network. Branching, edge-crossing, and nearby edges

caused the users attention to be distracted from the correct path [66] and edges incident to a node that go in the direction of the target of a path are searched first [67]. The latter finding contradicts previous work that found users tended to follow paths with the fewest turns [138]. There are currently too few controlled studies from which to derive a general theory of aesthetics and the experiments are often limited to only local, topological tasks.

2.2.2 How Networks are Understood

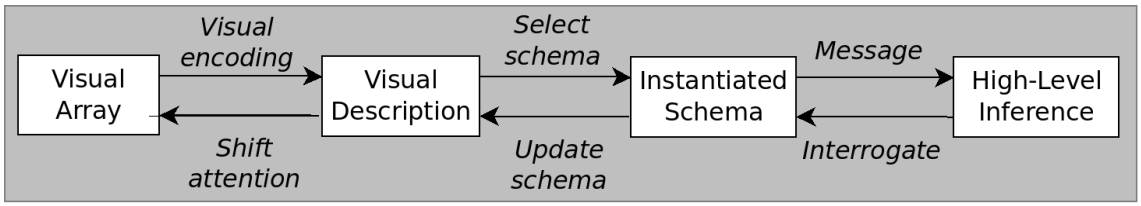

Figure 2.2: A brief outline of Pinker's "Theory of Graph Comprehension." [94]
Visual Array The image as perceived as patterns of light and dark on the viewer's retina.
Visual Description A visual encoding process structures and constrains the visual array according to perceptual principles (Gestalt grouping, representation of magnitude, etc.) to generate a default visual description. What is initially perceived is limited to a few highly salient items: an elaborated visual description is generated only after selective attention is applied by the schema. The encoding process may be primed to recognize particular patterns, hence experienced users can recognize significant features more rapidly than beginners.
Schema A given visual description is compared (in parallel) with schemata in long-term memory to find the closest match. A *schema* is a memory representation that can interpret a particular type of graph, and interface between the visual description and high-level inference processes. In response to interrogation by inference processes, the schema translates the request from a conceptual question to a visual query and locates the required visual information, (possibly by deploying attention in a visual search, or interactively by navigating to a new view). Once the result is supplied via the visual description, it is translated into a conceptual message and supplied to high-level inference processes. Responses will be fastest if the information was available in the default visual description, without requiring the viewer to search or navigate for the information.
High-Level Inference Inference processes interrogate the schema for new information by asking conceptual questions, and act in response to conceptual messages received from the schema. Given sufficient importance, certain inferences may ultimately be learned (stored in long-term memory). (Figure adapted from [94]).

In his "Theory of Graph Comprehension", Pinker [94] draws on evidence from visual perception and cognition to give a systematic account of how graphs are perceived and how visual forms are incorporated with high-level reasoning about the data. Pinker's

General graph schema	General map schema	General network schema
Pictorial content is linked to frame position via scales	Theme is linked to geographic position via geographic coordinates	Graph is linked to network position via embedding (layout) coordinates
Objects + parts described in terms of visual variables	Objects + parts described in terms of visual variables	Objects + parts described in terms of visual variables
Ratios-magnitudes specified in terms of a coordinate system defined by the graph	Symbol referents specified in terms of explicit or implicit assignment in a legend	Relations specified in terms of nodes connected by edges
Text grouped with objects labels or specifies absolute value of object	Text grouped with objects labels or specifies absolute value of object	Text grouped with objects labels or specifies absolute value of object
Relative position of objects specifies relative position in attribute space	Relative position of objects specifies relative position in geographic space	Relative position of objects is arbitrary

Table 2.1: General schemata for graphs (i.e. numerical plots) [94], maps [78], and a proposed general network schema. A general schema is an internal representation that contains the knowledge required to recognize and work with a given class of diagram.

model is summarized in Figure 2.2. Though Pinker's theory refers to diagrams that have a meaningful coordinate system, in contrast to the imposed space found in networks, the limitation only results from his choice of general graph-schema (note that Pinker uses the term graph in the general sense of numerical plot, rather than the specific meaning used in computer science). A general schema contains the knowledge required to recognize and work with a particular class of diagrams. Based on Pinker's theory, a general schema for recognizing and using maps was proposed by MacEachren [78] which he refines to include sub-schema for more specific representations and tasks. MacEachren's general map schema is reproduced in Table 2.1 and compared with a proposed general network schema.

2.2.2.1 Internal Representations and Schemata

The structure of internal representations of space are key to understanding users' conception of external space. Pinker shows how the visual description and schema can be described in propositional form (Figure 2.3 for example). MacEachren argues that, although Pinker's form contains spatial concepts (orientation and relative position for example), meaning is only attached to those concepts through the specification of the formal language itself [78]. He goes on to say that *image schemata* on the other hand are

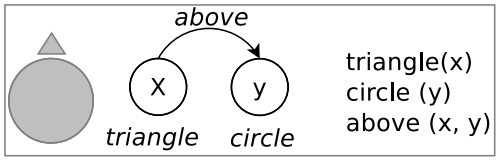

Figure 2.3: An example of a propositional form of graphical encoding from Pinker's theory of graph comprehension. Though expressive, the complexity of such a notation for even the simplest scenes can be seen as evidence that an image-based internal representation of visual stimulus is a more logical format.

inherently meaningful and there is neuro-physical evidence that map-like structures are used and retained in the neo-cortex. MacEachren also describes an *event schemata* that contain sequences of actions for achieving particular goals.

In relation to dynamic networks that change over time Purchase suggested "preserving the mental map" by fixing the positions of nodes was important to promote *object-constancy*, the ability to recognize a graphical object as representing the same item [99]. It was later suggested that beyond nodes, recognition of groups of features and user's ability to track moving targets required a more sophisticated notion of mental map [105]. Some potential types of features (symmetrical and orthogonal arrangements) were recently identified as being more memorable [81].

Meaning is attached to abstract spaces (maps or other types of visualization) using metaphors of our experiences in the real world: in schema terms, this suggests that we may interpret a visualization using schema derived from those used to interpret and navigate in more familiar, embodied situations [78, p. 196]. Studies of how people remember and navigate in real-world spaces may reveal potentially useful insights, on the principle that schema for navigating in abstract spaces are derived from more familiar, everyday knowledge. Tversky [127] suggests that rather than a map-like structure, internal representations of space are better thought of as a "cognitive collage". Collages comprise various representational media such as imagery, plans, and verbal-propsitional knowledge, arranged in overlapping and partial hierarchies, the structure of which is not a precise analogue of the real-world, being incomplete and systematically distorted [127]. A coarse representation is sufficient for navigation since corrections can be made as one progresses. The cognitive collage is similar to MacEachren's view that a mixture of propositional, image and event schema types are linked and instantiated in response to perceptual cues and guided attention mechanisms.

Understanding the distortions people make in their internal spatial representations can

lead to more effective external visual displays. A good example of how theories of comprehension and cognition can inform the computational design of a visualization system is the route-map system of Agrawala and Stolte [2]. Based on the systematic distortions people employ when giving directions, they devised a set of drawing heuristics: long roads were shortened and minor bends removed on the rationale that these features are of little informational value; and, angles of road junctions were increased to make them easier to perceive as when wayfinding along roads it is the junctions that provide the most useful information since they represent decision points. Reference points or *landmarks* that gave confirmation of progress such as bridges or junctions were preserved where they did not interfere with junctions. Agrawala and Stolte's maps were deliberately structured to reflect the structure of the user's internal representation, a property Tversky [128] refers to as *congruence*. MacEachren (referring to both Pinker [94] and Bertin [14]) puts forward a similar theory that,

> "information displays will be most effective when the designer uses a logical schema to organize the display and the viewer employs an identical schema" [78, p. 210].

The theories and approaches outlined so far suggest a connection between how one's internal representation can inform tools for navigation in the real world, and that understanding and exploiting similar correspondences between internal representation and visualization structure can lead to a more efficient design. The notion forms the basis of my approach to the network navigation problem, and these theories are referred to throughout the remainder of this thesis.

2.2.2.2 Landmarks

Landmarks have often been proposed as a way of supporting navigation in large-scale, 3-D virtual environments. Quinn et al considered the *memorability* of landmarks and suggest that simple structures are easier to use due to their low information content [100]. Vinson [132] argued that the way we navigate in, and learn the layout of 3-D virtual environments is analogous to those processes used in the physical world. Vinson suggests the use of landmarks on the principle that they are essential when following a route and that memories of routes are formed by linking landmarks. Survey knowledge allows one to consider a space from any perspective and is formed as a result of navigation experience [127]. Adopting Lynch's classification of navigational elements in cognitive maps of urban environments [77], Vinson recommends including all five types of ele-

ment: paths; edges; districts; nodes; and landmarks. (Lynch uses the term landmark to specifically refer to objects that can not be entered such as a statue).

In proposing a definition of landmarks suitable for both “real and virtual spaces”, Sorrows and Hirtle [116] criticized Lynch’s classification as considering almost every feature as an equally viable landmark. Instead they defined three dimensions that a landmark should contain, and suggest that the type of landmark to use depends on the type of navigation task.

Visual dimension relates to how visually *striking* the landmark is;

Cognitive dimension relates to how *meaningful* a landmark is; and,

Structural dimension relates to the *significance* of the location of the landmark.

Visual and structural dimensions are more useful in initial exploration, while navigating to a known target uses visual and cognitive landmarks. Sorrows and Hirtle [116] also highlighted an interesting sub-task referred to as *digression* where landmarks along a route are remembered as being of potential candidates for later exploration. These findings will strongly inform the design of the navigation components proposed in later chapters of this thesis.

While the evidence for virtual environments supports the general principle that there is a strong link between navigation in real and virtual spaces, it is important to remember that the user’s point of view in a network is not necessarily that of walking in an environment. If viewing a network is more similar to the aerial view of reading a map, there is evidence to suggest that survey knowledge is gained with reference to landmarks and need not include route knowledge [102].

2.2.3 How Networks are Imbued with Meaning

Text labels located close to elements in a network can associate specific meaning with those elements and the elements themselves can be rendered as colours and shapes whose perceptual properties immediately convey values or categories, referred to as “retinal variables” [14]. For this thesis the scope is limited to plain, unembellished networks yet even the spatial arrangement of nodes and edges can infer meaning. In networks, Dengler [32] showed that conventions about the semantics attached to different alignments of nodes were the same for both experienced and new users (Figure 1.6). Huang

presented evidence that nodes that are above or centred in a group are perceived as more important [68] in social networks. Dwyer et al [37] found that ensuring the distance between nodes was equivalent to the graph-theoretic distance produced networks that were "preferred" by users, though users often preferred symmetry over a more effective layout for some tasks. Recent empirical evidence showed that users were more likely to notice and remember symmetrical or orthogonal arrangements of nodes than other arrangements [81]. Layout algorithms based on aesthetics may fail to convey the meaning expected by the user as discussed in Section 1.2.1.3: there is a trade-off between concerns of aesthetics with those of semantics.

2.3 Navigation and Interaction

A taxonomy of graph tasks was drawn up by Lee et al [74] to support designers, evaluators, and to identify the strengths and weaknesses of existing tools. They identified seven objects of interest to users: graphs; nodes; links; paths; connected; components; clusters (spatially close and connected subgraph); and, groups (nodes with a common attribute). Four groups of operations are listed that involve those objects and show how analytic tasks can be composed of these operations. The groups are: topology-based; attribute-based; browsing; and, overview. The browsing group is of particular relevance to this thesis since it is composed of two navigation tasks:

follow path requires identifying and tracing sequences of links between adjacent nodes, referred to as *topological navigation* [85]; and,

revisit object is not tied to topology and can be thought of as navigating the substrate of the network.

I revisit this distinction in Chapter 3, where network navigation is discussed in greater depth.

2.3.1 Cognitive Costs

Though the task taxonomy described above links topological tasks with the user's intent, different solutions to navigation may require a variety of interactions by the user. Lam conducted a survey of information visualization evaluation literature to find reports of

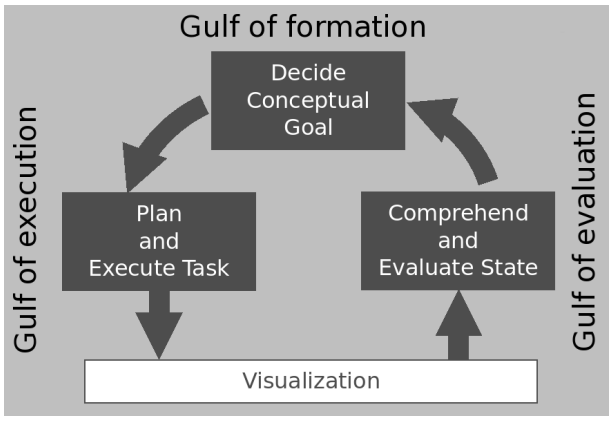

Figure 2.4: Lam's cognitive costs of interaction framework [73], based on Norman's "Seven Stages of Interaction". Poor interface design can cause additional cognitive effort, leading to three gulfs:
the *gulf of formation* involves costs related to forming high-level, conceptual goals;
the *gulf of execution* relates to the cost of translating conceptual goals to system commands; and,
the *gulf of evaluation* includes issues around perception, interpretation, and integration of the visual display with working memory. (Figure adapted from [73]).

the *cognitive costs* of interaction [73]. Based on Norman's "Seven Stages of Interaction" [92] Lam organized these costs of interaction around a framework of three gulfs, described in Figure 2.4. As costs accumulate, the gulf between the users's intent and the interface presented by the visualization system grows, leading to errors and confusion. Lam highlights seven specific areas where interaction costs can accumulate, described briefly below.

1. Cost of forming decisions, e.g. where to explore now?

2. Cost of selecting from many system operations.

3. Cost of detecting the current system operation mode.

4. Cost of physical movements.

5. Cost of visual clutter.

6. Cost of maintaining object-association between animated or multiple views and cost of local-global object association.

7. Cost of visual state changes that may prevent revisiting a previously found state.

Of particular relevance to this thesis are the object-association costs associated with maintaining visual references using a pan and zoom interface, the cost of visual clutter in comprehending the network view, and costs of forming decisions with regard to navigating the space of the network.

2.3.2 Solutions

Comprehensive surveys of various aspects of graph visualization are available. Herman et al [64] gives an overview of graph layout and detailed descriptions of interaction methods while more recently, Landesberger et al [134] highlight new interaction methods and discuss layout and graph-theoretic techniques more suited to very large networks. A comparative review of multiple-view, zoomable, and distortion-based interfaces for information visualization generally was carried out by Cockburn et al [26]. In the remainder of this section, a brief review of general network navigation techniques is given, followed by more detailed consideration of solutions related directly to this thesis.

Representation and layout options were outlined in Chapter 1. In this thesis, concerns of representation and layout are put aside in order to focus on how one navigates a given network. I assume therefore, that the representation is a node-link diagram with a layout already selected so they are not discussed further here.

Filtering reduces the amount of data to be displayed to a small subset with the advantage that the final view is less cluttered. Constellation [89] limited the number of nodes in view to enable an easily perceived layout, and allowed users to incrementally navigate along the edges of the network, bringing a new region into view. For certain combinations of task and data other topological methods can be used. Networks that contain *motifs*, interesting subgraphs whose structures appear frequently in a data set, can be filtered by searching for and extracting all instances of a given motif [133]. Filtering by semantics ignores topology and selects nodes whose attributes (associated values) fall within a selected range. Network Visualization by Semantic Substrates [113] for example, displays multiple filtered views with each view containing an attribute range, and arcs between nodes in related views. While filtering enhances the perception of selected features they do this by abandoning local-global association altogether.

Distortion techniques distort the substrate of the network to allow more space for features of interest while retaining some information about the surrounding context. One possibility is to perform layout in an alternative coordinate system such as hyperbolic space [87], that when rendered in the Euclidean plane, more space is given over to features in the centre of the view. Sarkar and Brown [107] introduced fish-eye views of networks that magnify one or more regions of the view to provide multiple foci while surrounding context is scaled down. A continuous scaling function means that the change between focus and context regions is smoothed rather than abrupt. The main difficulty with distorted views is that the features may be unrecognizable, or even not visible at all, depending on where in the focus or context region they lie, and the weight of evidence is that this prevents any expected performance benefit [26]. One way to combat this problem is to ensure the scale of important features is sufficient to guarantee they are visible [90].

Overview and detail uses multiple views at varying levels of scale with the aim of locating the current detail view within the overall context. Multiple views use additional screen real estate and requires additional effort to form local-global association between distinct views [26]. It was also suggested that the maximum scale difference between views should be limited to allow better recognition of features rendered at different scales [21].

2.3.2.1 Clustered Graph Visualization

The approach taken in this thesis is related to the visualization of clustered graphs in that they both seek to derive a visual abstraction of the data that can act as an overall framework for navigation. Clustered graph visualization refers to a set of techniques for representing an hierarchical abstraction of the underlying graph, with clusters or subsets of vertices treated separately. Suitable abstractions must either be present in the graph data or induced algorithmically. Abello et al [1] generated a hierarchy of clusters using an algorithm that forms clusters based on the density of subgraphs. Their rationale was to direct users to areas in the network of greater structural complexity. Use of an algorithmically-induced hierarchy was required by the size of graphs involved, but in response to feedback the authors added the ability to annotate nodes, helping users to revisit specific areas of interest.

Clustered graph visualizations hide topological detail, so interaction tools are usually

provided to open or enter clusters for closer inspection [40]. The exploration process can be streamlined by having the cluster provide perceptual or other clues as to what it contains, often referred to as "scent" or "residue" [49]. Herman et al [63] addressed the simpler problem of tree visualization, using the Strahler metric to replace sub-trees with schema triangles, and applying the metric to edge-width as a visual hint to the complexity of sub-trees. Plaisant et al [95] provided a thumbnail representation that gives a low definition overview of the structure of clustered sub-trees. More recent work dealt with providing visual cues in clustered graphs, for example Balzer and Deussen [7] represented clusters using implicit surfaces that mimic the shape of the contained subgraph at various selected levels of detail.

2.3.2.2 Zooming

In zooming interfaces users manipulate the scale of the view, zooming in to perceive the fine details of individual nodes and edges, and zooming out to gain a coarser overview. Coupled with the ability to pan, this gives users complete freedom over their point of view of a network. Furnas and Bederson [50] provided a useful conceptual model for understanding pan and zoom interaction in their "space-scale diagram" and applied it to the calculation of shortest-path trajectories between points in zoom-space.

"Desert fog", the problem of having no visual cues on which to make a navigation decision, was introduced by Jul and Furnas [70] in an analysis of navigation in zoom-space. In contrast, "critical zones", are contiguous regions in the current view where zooming in is guaranteed to contain information. Jul and Furnas [70] presented algorithms that compute all the possible views that contain at least one critical zone, and display them as rectangular outlines in an overview. When no critical zones are visible, i.e. desert fog, the simplest recovery strategy is to zoom out until one appears.

A second problem related to pan and zoom interfaces is that of *temporal frame association* [73]. As the view changes the user must expend cognitive effort to track objects, particularly when the objects move or change appearance [26]. Animated pan and zoom smoothly transitions from one view to the next to allow the user to track objects without the effort of interacting [131]. Despite being a passive viewer, users form mental maps from watching animated transitions [26].

2.3.2.3 Topological Navigation

In networks the critical zones can be thought of as any view of the substrate that contains nodes/edges. *Topological navigation* limits movements to the nodes and edges of the network. Assisted viewpoint finding and data-aware pan and zoom techniques have received much attention in the 3-D visualization literature, though techniques specific to graph visualization have now begun to emerge, Ahmed and Eades 3-D graph system [3] being an early example. In 2-D space, techniques such as link-sliding [85] allow users to quickly move to adjacent vertices by selecting an edge to follow. This method has been shown to work well in applications where the semantics and topology of the network are closely related, such as the tree-like structures of genealogy diagrams where edges represent parent-child relationships [15]. In the CGV system [123] panning along edges was augmented by an on-demand preview of all adjacent nodes, including those that are not in the current view. Topological navigation was combined with a degree-of-interest function to show only the context around a specific node of interest, providing the means to iteratively explore graphs where an overview is too large to display [130].

2.3.2.4 Off-Screen Visualization

Like the perceptual cues provided by clustered graph visualizations, visual cues as to the existence and extent of off-screen information are widely used. The scrollbar is a ubiquitous 1-D example; 'sunken' sections represents an entire document, while the raised widget show the user's current position relative to the overall content. This idea inspired City Lights [141], a variety of techniques to project off-screen objects on to the screen borders to indicate parameters such as direction, distance and identity, in the context of a street map application for small-screen devices. The Halo method [12] and later Wedge [55] built on this, both draw a simple shape around off-screen features that protrudes in to the view to give a perceptual hint about the distance and direction to nearby points of interest. WinHop [93] is a method that combined Halos to indicate the existence of off-screen targets, with automatic panning to a selected target. As with the all the techniques discussed so far, the method was implemented to support only a small number of off-screen targets as the shapes soon begin to overlap and create additional clutter.

A more scalable approach to dealing with off-screen features is to place a *proxy* representation of each target on to a second substrate displayed around the edges of the

network drawing, an idea first proposed in EdgeRadar [56]. Frisch and Dachelt [47] extended the design space of proxy-based off-screen visualization in the context of UML diagrams. Under *radial projection* the proxy appears on a line drawn from the centre of the display towards the centre of the off-screen feature. They highlighted the problem that using the edge of the view restricts the proxy space to one dimension, hence proxies can easily overlap. Also there is a perceptual discontinuity at the corners of the display as the proxy flips from one edge to the other, in other words, the 1-D display space is distorted with respect to the projection space. An *orthogonal projection* is especially limiting, as a large proportion of the off-screen space maps to the corners of the display. They suggested rounding off the screen corners to reduce the effect of distortion and presented several methods for stacking and merging coincident proxies, effectively creating a 1.5-D space. The design space of off-screen visualization is discussed further, and a new technique is proposed that addresses issues of distortion and cluttering in Chapter 4.

2.3.2.5 Visual Levels

The creation of *visual levels*, overlapping surfaces or groups of objects that can be attended to individually [78], is an important facet in the design of cartographic maps. Following a hundred and fifty years of advances, the ability of cartographers to create maps with multiple layers of information has enabled high information densities without sacrificing readability, as can be seen in Figure 2.5. The effect has received little attention in the information visualization community, although it was recently suggested as a method for managing attention in visual displays [101]. In the related area of set diagrams, contributions often focus on accurate representation of set membership (e.g. [27]), but must depend upon user perception to separate the two representations (items and sets) based on hue, shape, transparency, or Gestalt grouping.

A transparent overlay is one way that visual levels can be created. An exploratory study of a system for a civil engineering application showed that users could efficiently differentiate between two visually dissimilar layers when the overview was drawn as a semi-transparent overlay on top of the detail view [29]. Stone and Bartram [121] considered the effects of transparency on black and white reference grids, with the aim of drawing grids that were “legible, but not obtrusive” (see Figure 2.6). They later showed that when colour was introduced, red grids were as salient as black grids, but blue grids were less salient, hinting at a complex relationship between transparency and colour [10]. Stone and Bartram [10] also suggested that *x-junctions* formed between the boundaries

Figure 2.5: Careful creation of visual levels allows the cartographer to overlay several classes of information while avoiding clutter. This figure shows a comparison of map-making technologies from c.1850 with a contemporary design at the same scale. (Copyright Ordnance Survey, used with permission).

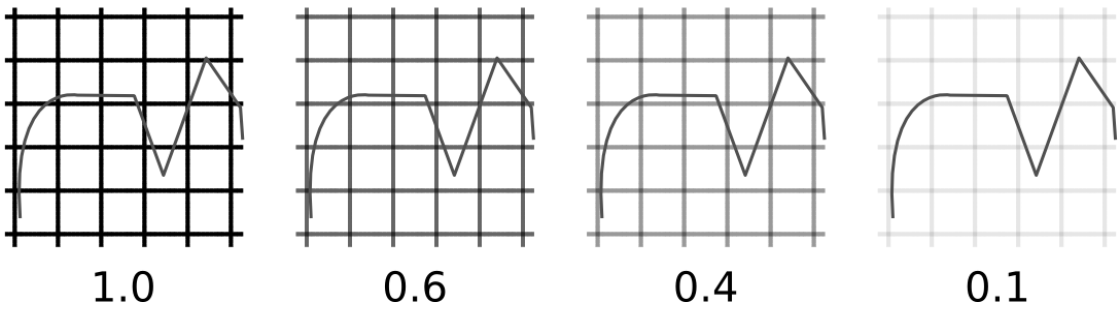

Figure 2.6: By testing reference grids rendered at varying levels of opacity, Stone and Bartram found that with an alpha of 0.4, the grid remains “legible, but not obtrusive”.

of transparent objects provide an important perceptual cue, allowing the visual system to resolve ambiguous overlap of transparent regions. Visual levels and transparency issues become relevant in later chapters, where the design of an overlay is presented as a means of imbuing a network with additional navigation information.

2.4 Software Tools

Software tools for network visualization can be seen as forming a continuum, from libraries that provide only data and algorithm structures with no direct support for visualization, through modular systems, to plug-in based and monolithic applications specif-

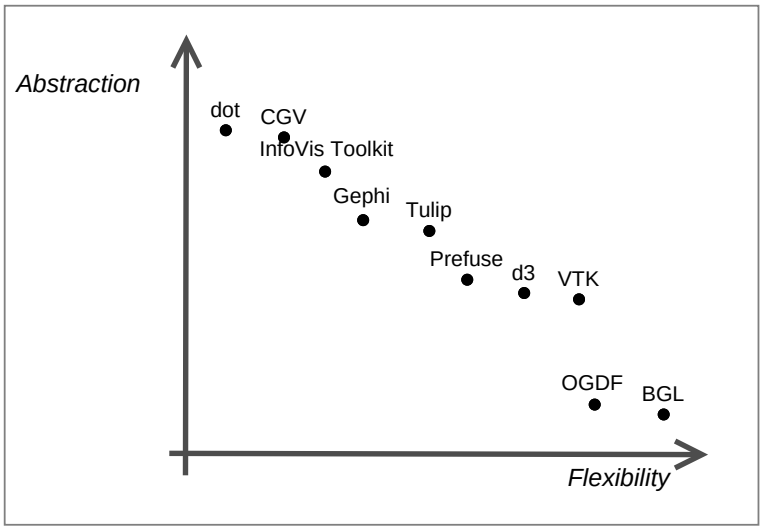

Figure 2.7: Examples of software tools to support network visualization implementation. The range spans fixed tools that though lacking in flexibility, can produce results quickly, to libraries of graph data and algorithm structures. In the middle lies a range of modular tools which can be quickly composed, and also provide support for customization.

ically designed for network visualization. Figure 2.7 shows a selection of exemplars. While basic graph libraries such as *Boost Graph Library (BGL)* [115] and *Open Graph Drawing Framework (OGDF)* [25] provide many useful graph-theoretic features and layout algorithms, a great deal of flexibility is afforded in how those structures can be represented visually. The trade-off for such flexibility is that the visualization designer must implement the graphics and interaction functions themselves. At the other end of the spectrum the static drawing package *GraphViz/dot* [52] provides a good selection of node and edge layout algorithms, in addition to a simple graphical language to create detailed designs, but lacks the architecture to provide efficient, dynamic interactions.

CGV [123] is a monolithic application that provides many network navigation and interaction tools 'out-of-the-box', leaving the user to decide which tool is appropriate for a given task. Simililarly, libraries of fixed visualizations such as *InfoVis Toolkit* [45] leave the choice of representation to the user, and are constrained to a small number of fixed designs. Network visualization specific frameworks, for example *Gephi* [11] and *Tulip* [6], provide a high-level application programming interface to provide data structures, algorithms and abstractions to support common interaction tasks. In this way these tools provide some ability to customize the design of an application through object-inheritance or plug-ins, but lack direct support for non-network visualization without extending the framework itself. Between these extremes lie modular tool kits for general visualization, including *Prefuse* [61], *Visualization Tool Kit (VTK)* [109], and *d3* [16]. The benefit

of modular tools is that they provide both a range of existing components that can be quickly composed to form a new design, while also providing the necessary application programming interface to create customized components.

VTK stands apart from the other tools mentioned in this section in that it was originally designed specifically to support scientific visualization. Despite the traditional distinction between scientific and information visualization there is evidence that the gap is closing. Duke [36] demonstrated that a scientific visualization pipeline could be adapted to support graph visualization and graph data types and other support for information visualization has since been added to VTK [139]. To support the research for this thesis I implemented a network visualization application using the VTK platform and added an overlay of navigation tools, the design of which is described in subsequent chapters. Chapter 6 contains a more detailed description of the VTK platform and I discuss my experiences of using the pipeline paradigm in an highly interactive visualization system.

2.5 Summary

In the first part of this chapter I reviewed the foundational principles of information visualization such as reference models and taxonomies, and found a recent trend towards a distributed approach where internal, cognitive models and external models of visualization are combined to provide a more complete picture. Towards a more thorough understanding of network comprehension in particular, I drew together perceptual evidence and cognitive theories from the domains of information visualization and cartography, and highlighted a connection between internal processes used in real-world navigation, and those employed in more abstract visual domains, suggesting landmarks as a potential method of improving congruence between internal and external representations.

In the second part some cognitive difficulties associated with navigation were identified, noting in particular costs of object-association, visual clutter, and, of forming navigation decisions. This led to a review of the existing solutions including clustering, off-screen visualization, and more recent work on topological navigation. I also considered the concept of visual levels as a method for adding additional overlays of information while avoiding clutter.

Chapter 3

An Overlay of Landmarks for Network Navigation

This chapter begins with a proposed model of network navigation based on the concept of landmarks. Implicit landmarks arise naturally from the arrangement of features in the current view, but lack the important properties of stability and semantic content. To counter these problems I propose a definition of explicit landmarks that may be overlaid upon the network to control the user's deployment of attention in a manner that is more efficient for navigation. Based on this model I go on to describe the selection and depiction of landmarks in order to maximize their utility.

3.1 Theoretical Model

3.1.1 Implicit Landmarks

As a starting point consider Lynch's theory of landmarks in urban environments, where every feature is a potential landmark [77], referred to in this thesis as an *implicit landmark*. In networks this equates to the possibility that at any one moment, a node, edge, or a group of nodes and edges, may act as a landmark, suggestive of a space containing combinatoric implicit landmarks. In scale-space the situation becomes more complicated, as the user modifies the view, different features become recognizable. Though scale-space is technically infinite, it can initially be bounded by the extremes of what the user is likely to view. At one extreme the entire network occupies the view (usually

referred to as an overview) and so zooming out further only serves to make the network appear smaller. At the other extreme, the user is unlikely to zoom in past the point where fewer than one or two edges are in view.

Furthermore, as suggested by Pinker's theory of graph comprehension [94], not every visual object within view is actually comprehended. Perceptual principles serve to limit the number of visual objects that become encoded and form part of the user's schema or internal representation. In particular, pop-out limits what is initially seen to striking features that contrast strongly with their surroundings. An hierarchical model of view decomposition [82] suggests that following pop-out, high-level features are recognized first, and access to individual items only follows if attention is deliberately directed to them. Therefore *salience* is likely to be a key factor in which potential implicit landmarks are actually encoded and used.

Unfortunately, the resulting implicit landmarks can not be guaranteed to assist the user in recognizing features, as the effects of graph layout do not take semantic information into account. Sorrows and Hirtle suggested a more refined model of landmarks in the context of both real and electronic spaces [116]. According to their model the implicit landmarks described so far are *striking* (i.e. salient) and *structural*, but to navigate successfully the landmarks should contain a *cognitive* dimension, with the benefit that top-down knowledge can prime the visual system to recognize objects that one would expect to see, given sufficient experience in a given domain. Priming of the visual system leads to faster recognition of significant features, and increases the likelihood that those features are encoded into the visual description, and hence available in the user's schema. Given sufficient exposure the availability of meaningful features may induce learning, adding the features into longer-term memory structures.

3.1.2 Explicit Landmarks

Due to the limited capacity of the visual system (only a few objects are available in the visual description at any given moment), and the uncontrolled way in which implicit landmarks may be recognized, it seems unlikely that useful navigational references will be comprehended by the user. Moreover, as the user moves around in scale-space, the set of landmarks available is subjected to frequent changes. Instead I propose that the addition of *explicit landmarks* in a controlled manner will provide a stable, and recognizable set of features, upon which to anchor the movements in scale-space.

To satisfy the requirements for explicit landmarks I suggest they must have the following properties:

Salience to ensure that attention is initially directed to explicit landmarks before the underlying network, thus overriding the encoding of implicit landmarks; and

Meaningful and significant so that user can more easily identify the conceptual content of features, and link them with their internal representation.

Note that a feature can be formed by more than a single node or edge, as *connected* subgraphs (or motifs) can represent expected configurations of individual relations. Therefore I define an *explicit landmark* as a connected subgraph that represents a significant or familiar unit of knowledge, overlaid upon the network with the aim of improving congruence.

In the remainder of this chapter I introduce the design of meaningful landmarks that aim to improve congruence: alignment of the user's internal spatial mental model with the meaning encoded in the graph. The alignment effect is achieved by highlighting explicit landmarks in the network. I have defined an *explicit landmark* as a connected subgraph that represents a significant or familiar unit of knowledge, in contrast to the *implicit landmarks* that result from the placement of edges and node-grouping introduced by the choice of layout algorithm, and while useful for navigation cannot be guaranteed to relate to the meaning of the underlying network. To create meaningful explicit landmarks there are two principal concerns:

- What is a landmark? That is, what features in the dataset meet our criteria for a landmark and, of these candidates, which are selected for representation?
- Given a well-principled selection of landmarks, how should they be depicted to maximize their operational utility? Specifically, how do we balance salience against occlusion of the primary network display.

These questions are addressed in the following sections.

3.2 Landmark Selection

To serve the dual purposes of *significance* and *utility* an appropriate choice of landmarks depends on several factors.

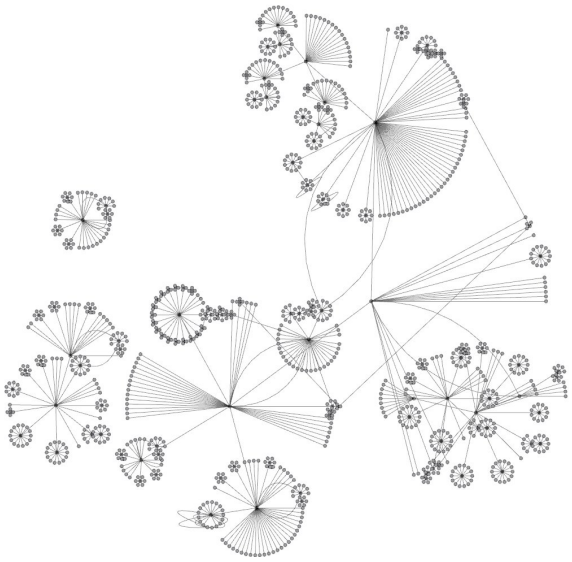

Figure 3.1: A network without explicit landmarks. Attention is guided by the salience of features that arise from topologically-based graph layout.

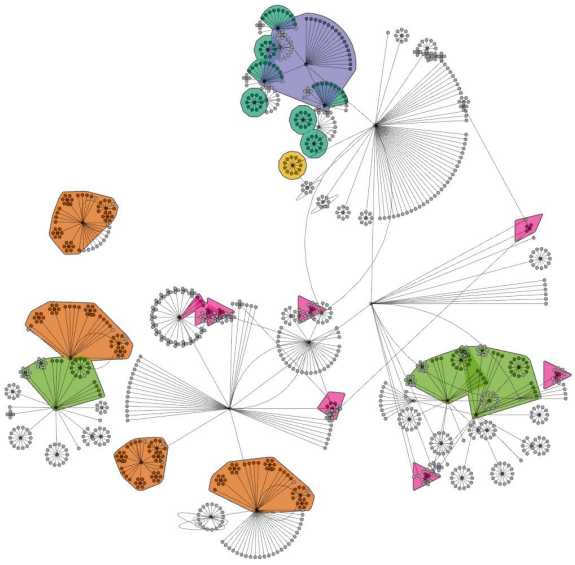

Figure 3.2: A network with explicit landmarks. The landmarks override the salience of the underlying network, guiding attention to significant semantic units within the network.

Domain What are the significant features in the domain?

Network semantics Of the domain-significant features, which appear in the dataset and how frequently?

User Of the features available, which are likely to be most useful to the user? For example, are there at least some features which will be immediately recognizable to the user?

Analysis task Does the nature of the user's analysis task require certain features to be made explicit?

Network size Clearly, the larger the network, the more landmarks can be included. A balance should be sought between too many landmarks which reduces their individual significance, and too few so that their effect is only active in a small part of the network.

Network layout The layout of the network directly influences the placement of landmarks. Some consideration should be given to how the resulting visual landmarks are distributed throughout the network and, where possible, avoiding the two extremes. All landmarks concentrated in one area reduces the utility of the landmarks by restricting their physical reach in the network. A uniform distribution of landmarks across the network increases cognitive load by limiting opportunities for "chunking" of landmarks: a group of landmarks itself forms a recognizable landmark, helping to distinguish larger regions of the network.

There is no algorithmic method to match the salience of features to specific tasks, and given the general nature of the landmark selection problem it is not possible to be prescriptive but a set of heuristics may provide some guidance. To summarize therefore, an ideal choice of landmarks would:

- provide a high-level semantic summary of the network;
- link useful and recognizable features in the network with the users existing knowledge;
- be 'nicely' distributed over the network display, and;
- be neither too few nor too many.

These guidelines are subject to a trade-off, for example one should not sacrifice an aesthetically efficient network layout in exchange for an ideal layout of landmarks. For

example, if the nature of the domain/task is such that the landmarks are concentrated in one area then this in itself may provide a fact about the content of the network.

In the following paragraphs I describe four possible methods of selecting landmarks and which to use is largely determined by the properties of the graph data. Motifs and metrics rely only on the topology of the graph, which in the case of a homogeneous structure such as a lattice, balanced tree, or fully-connected graph, gives little insight into local regions of the graph (all subgraphs have a similar topology). Semantic analyses depend on the availability of suitable attributes on the nodes and edges of the graph.

3.2.1 Motifs

Network motifs are defined as "patterns of interconnections occurring in complex networks at numbers that are significantly higher than those in randomized networks." [84]. Motifs represent the "basic building blocks" of the network they are drawn from. Analysis using motifs has been successfully applied to many phenomena that can be modelled as a graph, such as gene-regulation networks, molecular composition, and the hyperlink structure of the World Wide Web (again see [84] for a longer list of examples) where the topological structure of the graph provides useful insights. Landesberger et al [133] introduced a network visualization system where motifs could be defined interactively and then used as a basis for searching, filtering and clustering the network.

3.2.2 Metrics

Metrics are distance measures calculated from the topological structure of a graph and may be applied to graphs, subgraphs or individual nodes and edges. An example is node-degree: the number of edges incident to a given node. For the purposes of selecting landmarks it is important that the chosen metric reflects important domain-related information. In a hypertext network for example, Mukherjea and Hara [86] select landmarks based partly on the in-degree of a node, on the basis that important web-pages have many hyper-links pointing to them. Once metrics have been assigned to the elements of the graph a threshold may be used to select those elements which will be used as landmarks.

3.2.3 Semantic Analysis (node and edge properties)

Clearly some form of domain analysis and model is a necessary prerequisite for any visualization that attempts to incorporate domain semantics. Data mining and machine learning techniques assist in automating this modelling process, just as they do in the generation of ontological models (see e.g. [80]). However in small domains it may be possible to conduct a manual analysis to generate an appropriate set of candidates. I describe an example of such a process in Section 5.1, taken from earlier work in the specific domain of software memory visualization [91].

3.2.4 Annotation

Annotations are graphical marks added by the user, saving working memory resources by replacing the cognitive effort of remembering with a simple visual search task. The annotations serve to locate "chunks" of knowledge in the display, and provide common points of reference that can be easily revisited or shared amongst collaborators. In recent years there has been a trend towards architectures and standards to support the analysis process by specifically aiding the discovery, recording and coordination of analysis artefacts. For example a W3C recommendation [106] defined extensions to web services schemata that allows items to be imbued with meta-level semantics by linking to ontologies. Shrinivasan and van Wijk [114] proposed a framework to support visual analytic reasoning that defines three separate views: a primary data representation; a visual representation of the exploration process; and a knowledge view to coordinate analysis artefacts. The landmarks as described in this thesis can act as annotations that are overlaid upon the primary data representation to form a combined view.

3.2.5 Hybrids

Annotation is a useful adjunct to the previous methods with for example, metrics providing an initial spatial reference framework for exploration, and annotations being added during exploration. Other hybrid methods are possible, in the hypertext example cited earlier [86] a composite formula is used to take account of structural importance in addition to the attributes of individual nodes with the resulting landmarks being used to select and highlight areas of significance. A further possibility is to take a user-defined

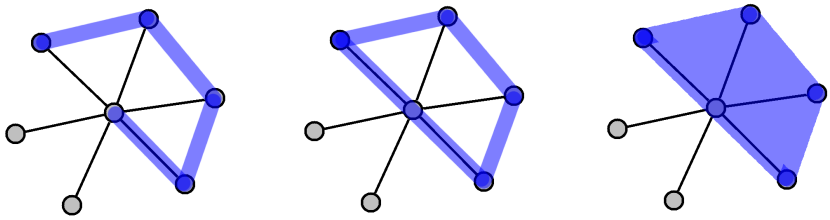

Figure 3.3: Styles of drawing contained sets. Lines joining set members have a similar appearance to network edges and introduce an ordering of nodes (left); contour lines avoid the ordering effect (centre); filled contours avoid ordering effects and provide good visual contrast with the network edges to form a separate visual level (right).

annotation as the definition of a motif (structure) or landmark class (structure plus labels) and automatically highlight other instances of that class in the network [133].

3.3 Landmark Depiction

The primary design goal is for the overlaid landmarks to be *salient* (to promote rapid acquisition) while *avoiding occlusion* of the underlying network. Examples of visualizations that highlight subsets of nodes in a network are common [5, 20, 27, 103, 118]. These works focus on accurate set containment while attempting to maintain interactive running times and use contour lines, filled contours, or joining lines to indicate set membership (see Figure 3.3). Of these methods only filled contours avoid becoming confused with edges of a network and provide sufficient shape and colour contrast with the network to form a *visual level*. The existing designs avoid occluding the primary display by placing the landmarks underneath the primary display with the effect that in a network with many edges, the contours themselves become occluded with the result that salience is lost. The use of a semi-transparent overlay and careful attention to the creation of visual levels as described here allows the contours to be drawn on top of the network without causing occlusion.

Since the purpose of landmarks in this case is to merely draw attention to the correct region of the display I relax the requirement for strict set-containment. I also use an overlay so that the landmarks may remain visible regardless of the density of the network and use transparency to allow the network to be read. Visual levels are formed as the overlay contrasts with the underlying network in both shape and colour, allowing the user to attend separately to the network or the landmarks.

The purpose of explicit landmarks is to locate meaningful subgraphs in the display. The visual encoding therefore must indicate both what and where the landmark is, more formally:

identity the knowledge artefact that is represented by the landmark; and,

membership the graph elements that comprise the landmark.

3.3.1 Identity

Textual labels alone give an approximate indication of landmark location but do not convey membership, size, and shape. Nor do they form a visual level, being easily confused with vertex or edge labels. Instead colour is used to either categorize or order the landmarks depending on whether the landmarks were selected by class or by metric value respectively. BrewerColor [18] provides convenient sets of colours for this purpose with variation in hue suitable for categorized landmarks and variation in saturation denoting an ordered set.

3.3.2 Membership

Drawing a filled contour around members of a subgraph is subject to a number of problems in general. One must avoid including non-members despite the vagaries of the graph layout algorithm. And where the subset is disconnected, additional marks or colour-based encoding must be introduced so that the set may be perceived as a whole. By restricting landmarks to a connected subgraph the latter issue is removed entirely. In the case of the former, I relax the requirement for strict set-containment: the purpose of the landmark is fulfilled even if non-members are accidentally included since its members are located and attention is drawn to that region of the network. A hierarchical theory of display decomposition [82] suggests that attention is first deployed to the top-level visual structure, the constituent parts only being attended to if the group becomes the focus of attention: for visual search tasks the top-level objects are key. Therefore I take the simple approach of drawing the filled convex hull of the landmark's vertices.

In early prototypes of the design I rejected two other approaches: 2.5-D tubes drawn along the edges contained within the landmark show accurate set-containment but provide little salience (they simpler cover fewer pixels) and were of no use in the case of a

landmark containing a single vertex; and, implicit surfaces were found to be relatively slow to compute and required careful parameter selection given the wide variation in the bounds of the input coordinates (i.e. the output of different layout algorithms). Recent implicit surface techniques such as BubbleSets [27] provide a near-linear approximation that correctly contain each set however, such methods require the entire network to be rendered. The simplified approach I describe here only requires the position of member vertices as input with the advantage that filled hulls can be calculated in an early stage of the visualization pipeline, immediately following graph layout.

3.3.3 Visual Levels

The overlay is rendered with alpha-blending to reduce occlusion of the network and to assist in creating the visual layer effect. Following Stone and Bartram's advice [121] on rendering reference grids an opacity of 40% was initially used, though this was found to be insufficient for the lighter hues with 50% being more easily perceived. With respect to the use of colour and transparency together, Stone and Bartram suggest there is a more complex interaction [10], and it was noted that landmarks in the overlay did not have equal contrast with the substrate. A useful advance here would be to adjust the alpha value of each hue so that they appear equally salient.

3.3.4 Guaranteed Visibility

Prerequisite to the provision of salience is the issue of visibility: the landmarks must be drawn sufficiently large for them to be seen. A problem highlighted in Munzner et al's TreeJuxtaposer [90] is that graphical objects may be culled by the rasterization process if they subtend less than the size of one pixel. Occlusion may also occur such that fragments of the landmark may not be written to the frame buffer. To overcome this TreeJuxtaposer artificially scales landmarks so that rendering to the frame buffer is guaranteed.

Beyond ensuring that the landmark is actually rendered to the display several additional factors must be taken into account to ensure that the mark is large enough to be seen:

- the density of the network as edges and nodes 'compete' for salience;
- the pitch and resolution of the display, i.e. the physical size of each pixel; and,

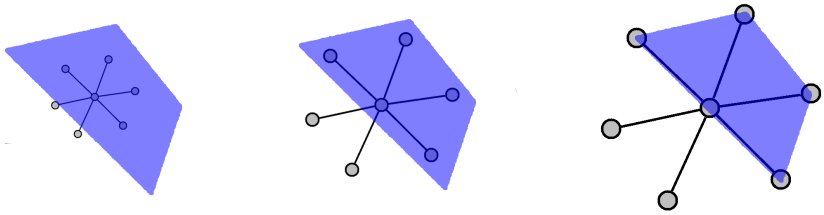

Figure 3.4: By fixing a minimum size in pixels for landmarks, they remain visible at any scale. In this figure the same landmark is shown at three different levels of scale.

- the visual acuity of the user (in concert with environmental factors such as lighting).

Where necessary we correct for this problem by scaling landmarks to guarantee a minimum-size bounding box (measured in pixels in both *x* and *y* directions). The landmark retains its original centre position and shape so it is still identifiable. As the user zooms in, any distortion is reduced until the on-screen dimensions of the landmark equal the required minimum. The minimum size is set initially to 30 pixels for a standard resolution 1680 x 1050 desktop display (approximately 3%), and can be configured interactively according to the user's individual preference. One case remains problematic: long, thin convex hulls are prone to invisibility and may appear excessively distorted when scaled.

One further problem occurs when the display can be panned as landmarks may no longer appear in the display as a result of frustum culling. This issue is dealt with in Chapter 4.

3.4 Landmark Management

The CoronaScope application includes a user interface so that users may activate sets of landmarks in response to changing task demands. A colour is associated with each set of landmarks and to assist the user in choosing sets of colours that provide a perceptually efficient range, several palettes from the ColorBrewer [18] tool are provided. Broadly there are two types of colour palettes: varying hues are used for nominal data; and, varying saturation is used to indicate a quantitative range. In the former case, hues that are easily separated and consistently named are used, to assist the user in remembering the otherwise arbitrary relationship between colour and class [137]. A property sheet displays the colours and the names of landmark sets associated with them, acting as a legend similar to those found in geographical maps (Chapter 6 contains further details).

In Lam's cognitive costs of interaction framework [73], she considered the need to support *reflective cognition*, allowing users the time to compare and contrast different hypotheses about the data. Furthermore, Lam provides evidence that interfaces that support refinding a previous state in a visualization encourage users to explore more, and goes on to suggest that allowing users to save the state is an important design consideration. The CoronaScope toolbox provides the means to save and load sets of landmarks in XML format so they can be reused between sessions, and to support collaboration between users (again see Chapter 6).

3.5 Summary

In this chapter I began by considering a network in scale-space as a combinatoric collection of implicit landmarks, constrained by the limitations of low-level perceptual mechanisms including the deployment of visual attention. To counter the problem that implicit landmarks in networks arise from topologically-based layout algorithms, and are unstable as the user moves around in scale-space, I proposed explicit landmarks with particular properties that ensure they are recognizable, and manipulate the viewers deployment of attention in a positive way. Based on the theoretical grounding I went on to set out several ways of selecting landmarks in a network that attempt to maximize the dual aims of providing useful navigational reference points and forming an explicit link between the user's existing knowledge of key high-level concepts in the domain and their location in the network display. I explored the design space of landmark representation in networks and suggest that coloured convex hulls provide the required effect provided the constraint of strict set-containment is relaxed. The use of shape and colours that contrast with the points and arcs of the network drawing creates a distinct visual level which, coupled with semi-transparency, avoids occlusion of the primary display of the network while maintaining a suitable level of salience.

Chapter 4

Landmark Awareness in Scale-Space

Simple, camera-based pan and zoom is useful in network visualization as a means to access both overview and the fine details of the network. The consequence of zooming in on a network is that, while a small area of fine detail is made readable, the global navigation context is lost. Significantly, any explicit landmarks added to the network may no longer be visible within the view since they are off-screen. To overcome this kind of problem *off-screen visualization* methods add navigational cues to the current view that indicate the presence and location of features elsewhere in the network. In this chapter I survey the design space of existing off-screen visualization, then propose a new technique that addresses problems of distortion and visual clutter. The new method, known as "CoronaScope", is then extended to provide specific support for two key tasks: following paths in the network that extend beyond the current view; and, revisiting features of interest.

4.1 Design Space of Off-Screen Visualization

Off-screen visualization refers to the set of techniques for providing visual cues that inform the user of important information that is not currently within the view. Such techniques are potentially useful in concert with pan and zoom so that when the user has focussed the display on one part of the scene, awareness of the global context can be maintained and it becomes possible to directly navigate to points of interest. Instances of off-screen visualization are common in document-based applications where distortion methods would make text unreadable. The Eclipse programming environment [41] for

example uses an enhanced scrollbar widget with annotation marks that indicate the position of significant features such as to-do comments and syntax errors. The annotations can then be used to rapidly navigate to the marked location.

In a survey of off-screen visualization designs very few methods specific to 2-D network applications were found so the scope was expanded to include a number of systems aimed at using urban maps on small-screen devices that include a pan and zoom interface and have a similar aerial view of the substrate. As positive results in user studies have been reported [56, 93] in the context of both networks and maps, the technique warrants further investigation. This review highlights three aims of off-screen visualization:

1. enable the user to *locate* an off-screen feature with respect to their current view;
2. enable the user to *identify* an off-screen feature without navigating to it; and,
3. be displayed in such a way that the main view is not obfuscated.

In the remainder of this section the existing design space is described in terms of these three aims. A summary of the designs reviewed (including references) can be found in Table 4.1. Screen shots comparing implementations of the main techniques are given in Figures 4.1 and 4.5.

4.1.1 Locate: Distance and Direction

Visually locating an off-screen feature from within the current view requires a representation of the direction and distance to some target. I identified two main methods for doing this:

1. draw a shape around the target that protrudes in to the view; and,
2. draw an object in the view that points to the off-screen target.

Amodal completion refers to the perceptual principle that a viewer may mentally complete cropped figures and interpret them as simple geometrical objects (see Figure 4.2). The Halo [12] system uses a circle centred about the target so the user must not only complete the shape, but also interpolate the centre of the circle to locate the target thus incurring significant cognitive costs. Wedge [55] simplifies the task by co-locating the off-screen point of a triangle with the target. In both cases, the viewable portion of

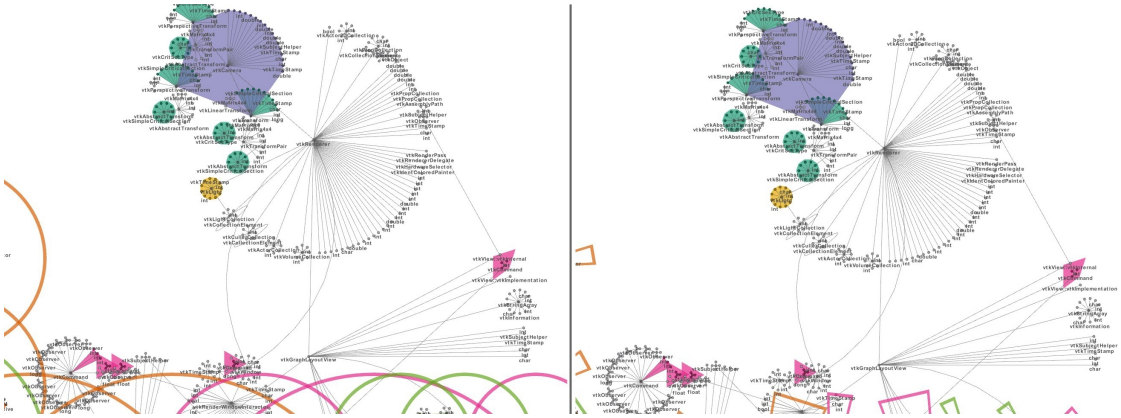

Figure 4.1: Comparison between amodal-based methods of off-screen visualization. Halo (left) and Wedge (right) locate off-screen landmarks using amodal completion.

the shapes can be compared with each other to quickly estimate the relative distance between objects. Figures 4.3 and 4.4 show how the amodal shapes are drawn with respect to examples of off-screen targets.

Proxy methods add items to the view that act as an on-screen representation of some off-screen target. The simplest method uses part of the edge of the screen as a scaled-down view of the off-screen parts of the network, though that method has limited scalability as the off-screen representation soon becomes too cluttered to discern individual items.

More sophisticated methods *project* the off-screen target into the view along a defined path. Figures 4.6 and 4.7 illustrate the idea, with a proxy being placed anywhere along the line of projection. The principle is that users can mentally reconstruct the line of projection and thus the path along which the off-screen target lies. An obvious limitation

Figure 4.2: The principle of *amodal completion*: simple, cropped figures can provide the anchor structure required to mentally project the hidden part of the shapes.

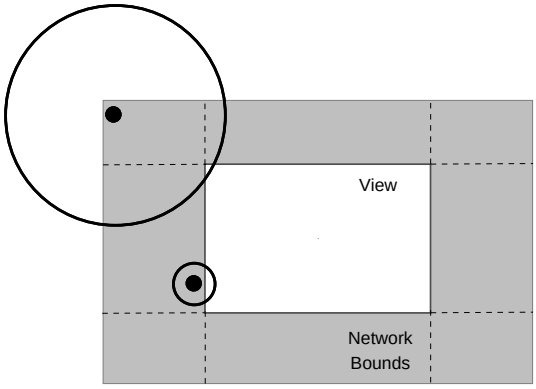

Figure 4.3: In Halo [12] a circle is drawn, centred about each off-screen target. This configuration requires the user to both project the circle *and* estimate the centre point. The white area in the centre of the diagram is the current view while the grey box represents the bounds of the network. Two off-screen targets are shown as black points.

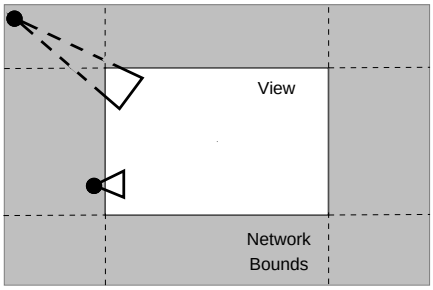

Figure 4.4: The Wedge method [55] uses a triangle whose off-screen point is co-located with the target. Unlike Halo in Figure 4.3, the user does not need to estimate the centre of the shape.

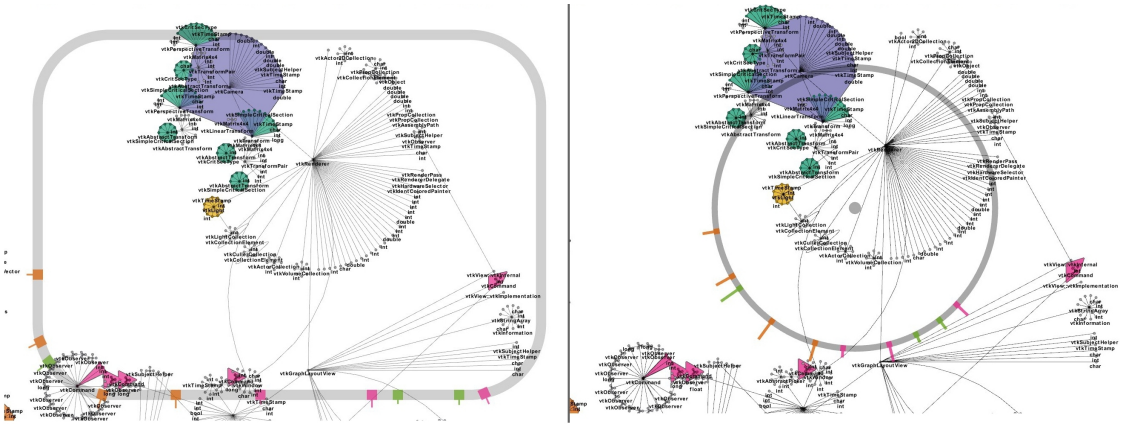

Figure 4.5: Comparison between proxy-based methods of off-screen visualization. BorderScope (left) and CoronaScope (right) use proxies that point to their off-screen counterpart.

of *orthogonal projection* is that there are significant areas of off-screen space whose on-screen location is undefined, the simplest solution being to dedicate the corners of the view to these spaces, though this can rapidly lead to clutter. Furthermore, the movement of proxies as the view is panned around the network is not smooth as proxies appear to 'stick' at the corners. *Along-edge projection* is a form of orthogonal projection used in networks where proxies are placed at the edge of the view, at the point where the graph-edge that links the off-screen target intersects the edge of the view. Frisch et al [47] explore this idea more deeply and present several schemes for dealing with the problem that the off-screen target is not necessarily in line with the edge along which the proxy is placed.

Under *radial projection* the entire off-screen space can be represented equally with no distortion, though there is still a tendency for proxies to clutter the corners of the view. The solution often employed in this case is to round off the corners of the view to provide more space to fit proxies in those regions.

4.1.1.1 Additional Distance Encoding

Some methods display only a limited indication of distance, for example RadarView [123] groups proxies around two concentric circles to give a coarse representation of near and far targets. In addition to encoding the distance to the target using the length or position of the proxy/amodal shape, some methods [12, 55] also reduce the opacity as target distance increases. Choosing this type of encoding over more obviously 'spatial' methods

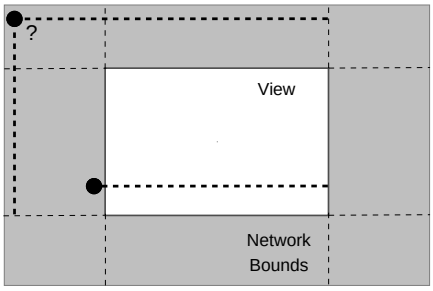

Figure 4.6: Orthogonal projection of off-screen features. A *proxy* of an off-screen target can be placed anywhere along the line of projection, pointing towards the target. Targets that fall within the regions of the network in the four corners of the off-screen area (shown here in grey) have no line of projection that intersects with the view.

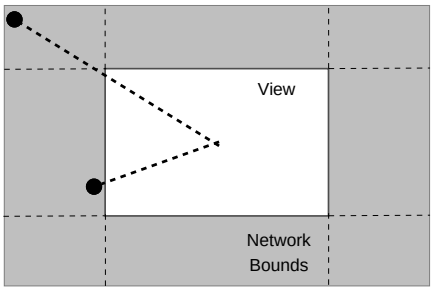

Figure 4.7: Radial projection of off-screen features. A proxy of an off-screen target can be placed anywhere along the line of projection, in the direction of the target. The origin of the line of projection can be at the centre of the view as shown here, or may be taken from the current cursor position. Using radial projection, all off-screen regions can be represented within the view.

seems counter-intuitive, adds complexity, and uses additional perceptual dimensions that could be used to encode other attributes.

4.1.2 Identity

Hue, text labels, and appearance (shape) have all been used to signal the identity of a proxy object. A significant advance was “Dynamic Insets” [53] that aid identification by providing an image of the off-screen target and its surrounding context. Each proxy was interactive, using the same modes as the main display so could be zoomed as required. The technique seems to work well in map-like applications since the high density of graphical marks usually provides sufficient context. However, it is not clear how this technique extends to networks as the availability of useful context cannot be guaranteed while keeping the intended target object visible at a reasonable scale.

4.1.3 Visual Design

In the amodal completion methods there is little flexibility in the overall layout, unlike proxy-based methods which allow some freedom over where in the line of projection to place the proxy objects. Also, under radial projection one must decide where in the view to project from, the centre of the view or, the current cursor position. The latter option is more suitable when the off-screen representation is presented on-demand as continuously moving and updating the location of the proxies or any reference structure could be confusing.

The space allocated to proxies can be limited to a 1-D space at or near the edges of the view. This means that only direction is indicated by the position of the proxy and some other perceptual dimension must be used to encode the distance to the target. Where a small space at the edge of the view is allocated to proxies a 2-D scaled mapping of the off-screen space soon becomes too distorted to be usable. This is less of a problem for those methods that provide an on-demand display since a larger proportion of the view may be used and the proxies can be targeted to a particular direction of interest [123].

A compromise is to stack overlapping proxies to create a 1.5-D space where the nearest proxies are nearest the top of the stack which deals with the problem of clutter but only shows the relative distance in the case of overlaps. This works well in the case that the Euclidean distance is considered less important, for example relatively small class

diagrams [47]. A further option is to encode the distance as part of the proxy itself using a retinal variable such as item-length. Many of the techniques reviewed used distortion, deliberately perturbing the position of the proxy to avoid overlaps between proxies.

4.2 CoronaScope: A Novel Off-Screen Visualization

Since existing off-screen visualization designs are limited to adjacent nodes or suffer from clutter and distortion around the screen corners a new design to support off-screen landmarks is desirable. This section describes a new technique that aims to address these problems and provide an effective method of indicating the presence of off-screen landmarks. As before, a key design constraint is to avoid occluding the primary display of the network. The new method is named “CoronaScope”, “corona” as it resembles the sun’s corona and “scope” being an instrument for viewing. Figures 4.11 and 4.8 provide schematic illustrations of CoronaScope’s main elements. Briefly, CoronaScope is composed of a circular *bezel* drawn in the centre of the view that provides a frame of reference or ‘visual scaffold’ around which *proxy* elements are placed. These components, and other features of the design are justified and described in the remainder of this chapter.

4.2.1 Bezel Design

A circular *bezel* is drawn in the centre of the view, providing a visual scaffold or reference frame around which the *proxy* elements are placed. The use of radial projection coupled with a circular bezel has two positive effects: off-screen targets are given equal space around the bezel regardless of their position with respect to the current view so there are no problematic corners where bunching can occur; and, the movement of proxies around the bezel is smooth and continuous. Furthermore, the organization of navigation cues around a central structure, rather than distributed between points in the display or around the border, reduces the perceptual cost of visual transitions [137] by placing the proxies close to the user’s expected point of focus. In addition to the bezel, the centre of the screen is marked by a small circle that assists the user to form a mental representation of the line of projection, starting at the centre circle and passing through a proxy in the direction of an off-screen landmark.

If the user is not interacting (i.e. not moving the mouse) then it may be reasonable to

	Direction	Projection				Layout			Corners		
	Amodal or Proxy	Radial (display centre)	Radial (cursor pos'n)	Ortho-gonal	Along-edge	Border	Circle	Area	No corners	Rounded	Square
Halo [12]	A	●				●				●	
Wedge [55]	A	●				●					●
Hop [69]	A	●				●	●			●	
WinHop [93]	A	●				●	●			●	
Class diagrams [47]	P				●	●				●	
Dynamic Insets [53]	P				●	●					●
Bring & Slide [15]	P			●				●	●		
RadarView [123]	P	●				●				●	
Bring & Go [85]	P		●				●		●		
EdgeRadar [56]	P			●				●			●
CoronaScope	P	●					●		●		

Table 4.1: Classification of existing off-screen visualization methods compared with CoronaScope, across key design attributes. CoronaScope uses a unique combination of radial projection and a view-centred, circular layout that avoids the problems of bunching and distortion in corners.

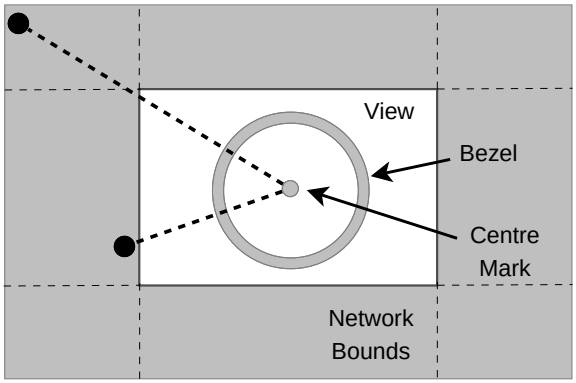

Figure 4.8: Off-screen distance and direction indication design used in CoronaScope. A centre-mark and bezel are rendered as an overlay. Proxies can be placed where the line of projection (illustrated by the dashed lines in the figure) between the centre-mark and the off-screen landmark intersects the bezel.

assume the user is viewing some details of the network and therefore not referring to the global navigation cues. This leads to a further design enhancement whereby the opacity of both the bezel structure and the proxies is reduced to a very low level (5%) when the navigation aid is not needed, causing the off-screen overlay to mostly fade out of view. Any cluttering effect is removed and the user can more easily attend to the network details. Retaining a faint trace of the overlay provides affordance, acting as a reminder to the presence of the navigation aid and thus reducing the cognitive costs of view-state changes (see Figure 4.9). Once the mouse begins moving again the original opacity level is restored. The overlay can be held in the visible state by hovering the mouse over the bezel, allowing the user to override the assumption encoded in the visual design.

4.2.2 Proxy Design

4.2.2.1 Distance and Direction

The main body of the proxy is a sector shape that moves smoothly around the bezel in response to changes in the position of the view, and acts as a visual anchor being large enough to be readily perceived. The proxy is located on the bezel such that it falls on the *line of projection*, a conceptual line that extends from the centre marker to the off-screen target. A pointer or *indicator* is added to the main body to provide more fine detailed information about the direction and distance of the target. Pointer marks such as arrows and lines are often used as graphical “gestures” [129], a metaphor for the

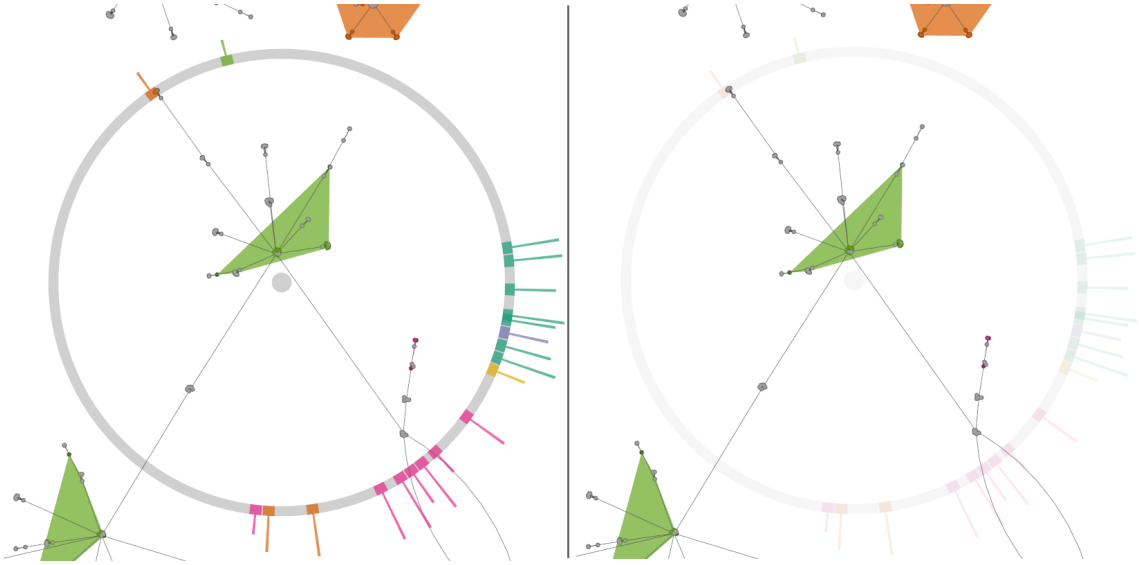

Figure 4.9: When the user is not navigating, the CoronaScope fades smoothly from 50% (left) to 5% (right) opacity, leaving a faint trace as affordance. Once navigation resumes, the original opacity is immediately restored.

dynamic hand movements people use to describe locations in space. Here the indicator is used to similar effect, pointing along the line of projection and towards the off-screen target. The indicator is scaled so that the length is proportional to the distance to the off-screen target. Distances to off-screen targets are calculated in world coordinates so that the length of a distance indicator is invariant to the current level of zoom, otherwise it would be necessary for the length of the pointers to represent an almost infinite range.

Limiting the length of the indicator is essential to ensure that it remains visible within the view. The *maximum* length of the distance indicator is calculated to fit within the smallest space available between the bezel and the edge of the view. Assuming the view contains at least some part of the network, the maximum possible distance between view and off-screen landmark is approximately the diagonal of the bounding rectangle of the network (see Figure 4.10). The *minimum* possible distance in world coordinates between the centre of the view and an off-screen landmark decreases as the view is zoomed in. To ensure the perceptual effect of pointing to the target remains, a minimum indicator length of 10 pixels is fixed: on a standard desktop monitor this is sufficient to add a noticeable protrusion to the smooth sector shape of the proxy (see Figure 4.11).

The limited space available to display the proxy indicators has potentially negative consequences since in large networks where high levels of zoom are required, off-screen

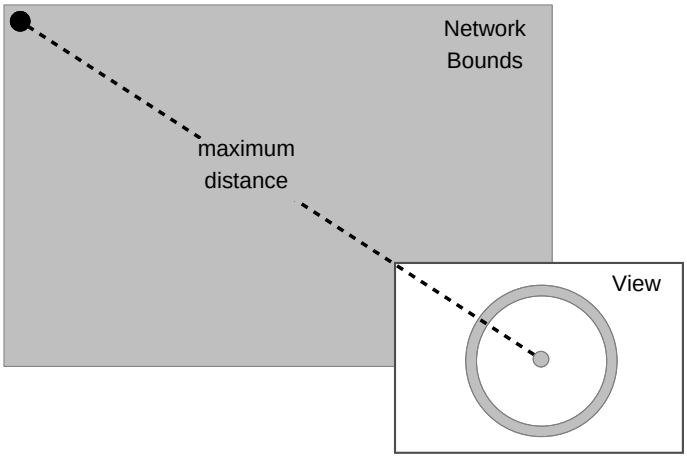

Figure 4.10: The maximum pointer length is calculated to be proportional to the maximum possible distance to an off-screen landmark. Assuming some part of the network is in view, the diagonal of the bounding rectangle of the network provides a reasonable estimate of the maximum distance that the user would wish to navigate.

landmarks that are near may not be accurately represented. Similarly, the range of possible distances is poorly represented by the relatively small number of pixels available to encode them. This loss of fidelity is a typical example of the kind of trade-off one must make when creating a layered visualization design: increasing the space available to indicators by reducing the diameter of the bezel increases the visual clutter of the primary representation of the network. Returning to the gesturing metaphor that led to the indicator design, its purpose is to provide an approximation of the magnitude and direction of the distance to a landmark.

4.2.2.2 Identification

To aid in identifying the feature referred to by a proxy, the proxy is filled with the same colour as its off-screen counterpart. Text labels provide a much more direct identification (since the meaning encoded by a colour must be remembered or looked up in a legend) but add a large amount of visual clutter, especially if the network nodes and edges are also labelled. To avoid the additional clutter, a text label of the landmark name is shown only when the user hovers over a proxy with the cursor.

In the context of navigation in virtual environments users relied on features formed by spatial configurations of landmarks such as an ‘L’ shape to provide directional information, though in that study the landmarks were uniform in appearance and placed randomly throughout the scene [31]. If the same spatial grouping affect applies here, it

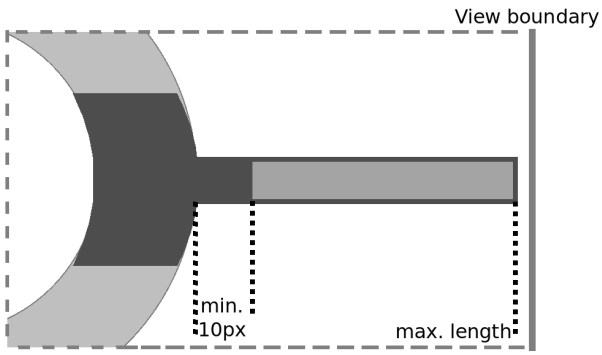

Figure 4.11: CoronaScope's proxy design. The main body of the proxy is sector-shaped so that fits neatly within, and moves smoothly around the bezel. A distance indicator grows and shrinks in response to pan movements to encode the distance "on the map", subject to a minimum size of 10 pixels to ensure the indicator is visible.

may aid identification of landmark instances if they are part of a larger spatial feature. The group of landmarks in Figure 4.14 illustrates an example where the group occurs as a result of a close semantic relationship thus the configuration of features is expected. This hints at the possibility of a hierarchy of landmarks, an idea that is returned to in Chapter 7.

4.2.3 Overlap Removal

As increasing numbers of off-screen points are projected on to the bezel, overlaps are inevitable as a 2-D space is projected into 1-D, since the proxies are fixed to the bezel. Overlapping proxies can be difficult to recognize and distinguish from each other, and selecting an overlapped proxy with the mouse can be similarly challenging. In particular, since they are transparent, two overlapping proxies can cause confusing blending artefacts and appear as three objects. My initial solution was to exploit x-junctions by staggering the baseline of the proxies, but this creates a more complex view (see Figure 4.12). Similarly, stacking the proxies or adding additional concentric bezels to represent increasing radial regions in off-screen space were rejected in favour of the perceptually simpler option of perturbing the proxies to remove overlaps altogether.

While there is no direct evidence of how users internalize off-screen space, studies of how people remember and recall real-world spaces revealed distortions in people's internal models [127]. This suggests that the requirement that proxies must indicate the *exact* direction to the off-screen point may be relaxed, allowing the positions of proxies to be perturbed so that confusing overlaps are removed. The error introduced by per-

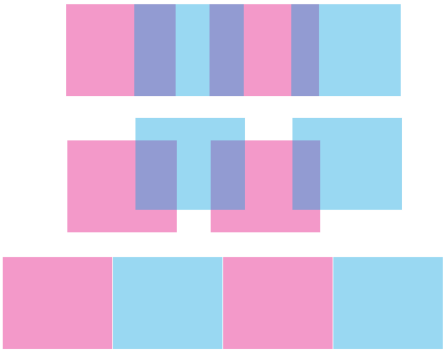

Figure 4.12: Overlapping, transparent shapes can create ambiguity as the individual shapes can not be easily distinguished (top). Creating X-junctions by staggering the shapes resolves the problem (centre) while perturbing the horizontal position creates a simpler view (bottom).

turbing proxies diminishes as the user navigates closer to a target point, with the positive effect that, for distant landmarks the proxy pointers convey a sense of direction, while for closer landmarks they provide finer spatial discrimination. A similar observation was made by Jul and Furnas in the context of zooming interfaces [70]. Although the apparent distance between off-screen landmarks may appear distorted, the relative position of landmarks is represented, provided the order of proxies around the bezel is not changed.

A related thread of research proposes algorithms for removing overlaps between rectangles in two dimensions, a common application being networks where rather than an infinitesimal point, nodes are represented by a significant shape and can contain a text label. Generally these algorithms proceed by setting separation constraints between objects then solving a set of linear equations to find a solution. Solving this optimization problem in two dimensions was shown to be NP-hard, so current algorithms often simplify the problem by first solving for the x-direction, then y, and merging the two results to provide a reasonable approximation. For example Dwyer [38] gives an algorithm that removes overlaps in near-linear time by moving the rectangles as little as possible, and a slower variant that takes into account the relative distance between objects so that grouping effects are preserved. Removing overlaps in two dimensions has the advantage that the space available on the plane is infinite whereas the proxies in CoronaScope have only one degree of freedom, radially as they move around the bezel. The constrained space means that rather than preserving relative spacing between proxies, I propose a simpler algorithm that produces an exact solution with the following characteristics:

- the radial ordering of proxies is preserved;

- the amount of distortion applied to any one proxy is minimal; and,
- the proxy positions can be updated within the limited time available to maintain interactive frame-rates.

Although distances between off-screen landmarks are not accurately represented using this method, the fixed radial order ensures that relative positional information is preserved. A detailed description of the algorithm follows.

4.2.3.1 Algorithm Description

A *segment* is defined as an ordered array of angles, each angle referring to the mid-point of a proxy, measured around the bezel with the origin in the centre of the view. The *arc* of a segment is defined as the amount of space required to distribute its constituent proxies, separated by δ, and the *mid-point* is the angle half way between the first and last proxies in that segment. Given the arc and mid-point of a segment, the expected left and right extents of the segment when overlaps have been removed may be calculated, since each proxy requires δ space, hence given a segment S containing n proxies:

$$LEFT(S) = \frac{S_1 + S_n - \delta n}{2}, \quad RIGHT(S) = \frac{S_1 + S_n + \delta n}{2}$$

If the product of the total number of proxies and δ is greater than the amount of space available around the bezel ($\delta n > 360$), then there is no solution, though this is unlikely in practice given the relatively small number of off-screen landmarks.

Initially, input is supplied to the algorithm as an ordered array of segments, where each segment contains one proxy. The algorithm then proceeds in two stages:

Merge overlapping segments In the first stage, consecutive pairs of segments are compared and if the expected extents overlap, the two segments are merged into one, such that the order of proxies within the new segment is preserved, and the length of the segment array is reduced by one. Note that the first and last segments in the array may also overlap so the first comparison is between the *last* and *first* segments in the array (it is assumed that calculations are performed using angles in degrees and are *modulo* 360). As the merge operation recalculates the combined arc of the two segments about their new mid-point, new overlaps with neighbouring segments can be created. Overlaps to the front are dealt with in the next step, as the new segment is compared with its successor. Otherwise, the entire merge

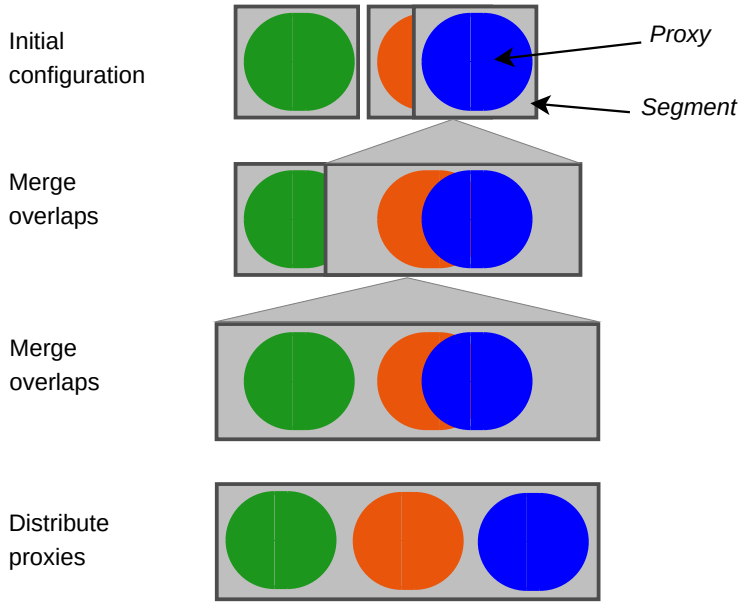

Figure 4.13: Illustration of the proxy overlap removal algorithm. In the first stage a segment is merged with its successor if they overlap. Once all segment overlaps have been removed the proxies are evenly distributed within their segment. The result is the removal of overlaps with minimal distortion to each proxy.

operation is repeated with the new set of segments until no new overlapping segments are found, or only one segment remains. The result is an ordered array of non-overlapping segments, where each segment contains an ordered array of overlapping proxies.

Distribute proxies The second stage takes each segment in turn and distributes its proxies equally within the bounds of the segment which, since the segments are non-overlapping, does not create any new overlaps. By distributing the proxies in the minimum amount of space required, the displacement of any one proxy is minimized. Figure 4.13 illustrates the process using a small example.

Pseudo-code is given in Algorithm 1, and a brief analysis of asymptotic bounds [28] follows. Let n be the number of proxies. In the first merge loop there are n comparisons (Line 2), and in the worst case, the n^{th} comparison detects an overlap and merges two segments, thus reducing the number of segments by 1. Let k be the number of iterations of the merge loop required to remove all overlapping segments. Since initially $k = n$, in the worst-case only one pair of segments is merged, and one new overlap is created in every iteration then there are at most

$$n + (n-1) + (n-2) + \ldots + 2 = \frac{1}{2}n^2 - \frac{1}{2}n - 1$$

comparisons to give an asymptotic upper bound of $\Theta(n^2)$. The example in Figure 4.13 illustrates the worst case: reversing the direction of the merge loop would find all the overlapping segments in the first iteration and thus solve the example in just n steps. As the segments are somewhat sparse then in practice only a relatively small number of new overlaps are created following each merge loop, therefore for most configurations of proxies, running time is close to $\mathcal{O}(n)$.

Algorithm 1 Remove proxy overlaps.

Input: P an ordered array of segments, where each segment contains one proxy.
Input: δ the required separation between proxies.

```
 1: repeat                                          ▷ Merge overlapping segments.
 2:     n ← |P|
 3:     previous ← P_n
 4:     for i ← 1 to |P| do
 5:         if RIGHT(previous) > LEFT(P_i) then
 6:             MERGE(previous, P_i)
 7:         else
 8:             previous ← P_i
 9:         end if
10:     end for
11: until |P| = 1 or |P| = n

12: for i ← 1 to n do                               ▷ Distribute proxies.
13:     for j ← 1 to |P_i| do
14:         P_{i_j} ← LEFT(P_i) + δ(j − 1)
15:     end for
16: end for
```

Output: P an ordered array of non-overlapping segments.

Figure 4.14 shows a comparison between the arrangement of proxies before and after overlap reduction. The figure also shows a further design iteration whereby the error introduced by distorting a proxy position is also displayed, a feature that was added in an attempt to replace the absolute distance information that is lost when proxy positions are perturbed. In order to clearly indicate that proxies have been perturbed the tip of the direction remains in its original position. The effect of encoding this additional information is that the main proxies give a coarse indication of the relative arrangement of a group of off-screen landmarks, while more detailed attention to the distance indicators provides a more accurate picture. Clearly this connotes additional cognitive effort, and since the structure of the user's cognitive model of off-screen space is not well understood in practice, caution should be used in adding perceptual complexity where the

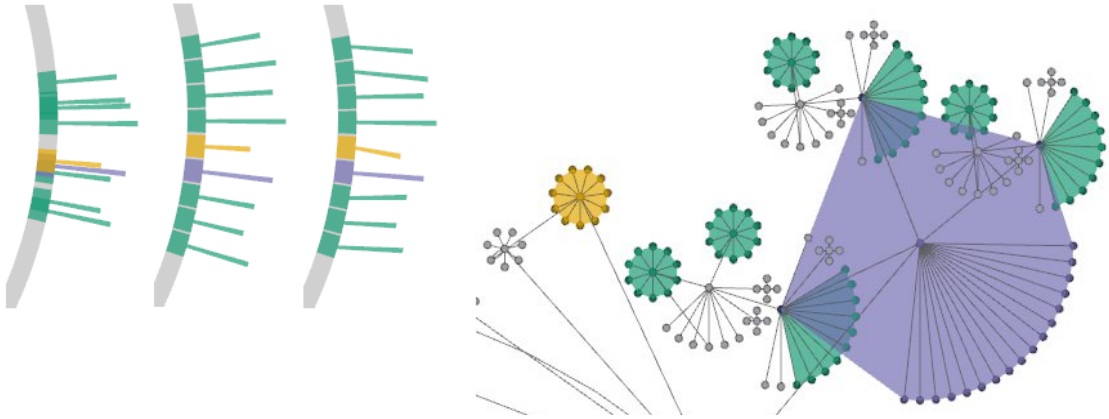

Figure 4.14: (Left) Overlapping proxies can be difficult to distinguish from each other. Since they are transparent, and lacking x-junctions, two overlapping proxies can appear as three. (Centre) After overlap removal the proxies retain their original ordering and are more easily discerned. (Right) The amount of distortion applied to each proxy is encoded by rotating the tip of the distance indicators back to the original positions. (Far right) The off-screen arrangement referred to by the proxies in this figure.

cognitive benefit is not clear. Lacking a firm theoretical justification for its inclusion, the feature was left as a user option.

4.3 Following Long Paths in Scale-Space

As well as being a fundamental component of many graph tasks [74], the ability to follow paths is *the* rationale for selecting a network representation [54]. Evidence from user studies suggest that following a paths of more than seven nodes in a network is particularly challenging [98, 138]: as the number of branches, edge-crossings, and bends increases, the cognitive costs of these decision points accumulates leading to task breakdown. Although there is no empirical evidence of path-following tasks that extend beyond the current view, and thus requiring pan and zoom interaction, additional cognitive costs of navigation arise from the need for the viewer to maintain temporal-frame association [73]. Also relevant are the finding that users tend to search for the next node in the path either in a straight line or towards the target [67], neither strategy is likely to be helpful in the case that the correct path meanders toward the final target.

Previous work on topological navigation has provided solutions to incrementally traverse a network by moving between adjacent nodes [85]. Having created a framework of global navigational reference points, it becomes possible to provide interactive tools

for navigating along long paths in scale-space. Briefly, a path is calculated between a node nearest the centre of the view and a node in a selected destination landmark, and an animated pan and zoom smoothly translates the view to the target. The target landmark is selected by clicking the mouse while hovering over its proxy representation in CoronaScope. The calculation of the network path, and the path of the animation are described next.

4.3.1 Network Path Selection

The start point for the network path is the node nearest to the centre of the view, which can be explicitly selected by positioning the desired node within the centre marker. After some initial testing it was noted that when the nearest node led to an edge that leads away from selected target landmark, the effect was that the animation initially moved away from the target, in a direction that was not anticipated by the viewer with the effect that they become temporarily lost. To avoid the jarring effect, the selection of a start node is constrained to within a 180° arc, centred in the direction of the target. The end point is simply the representative node of the target landmark.

Having determined the start and end nodes a graph-path can be calculated, which has one of three possible outcomes:

- the nodes are in distinct connected components and therefore no path exists between them;
- only one path exists; or,
- there are multiple paths that may be followed.

In the first case where no graph-path exists, the network edges are ignored and a direct line across the substrate between the start and end points is used. In the case that only one graph-path exists then the selection is trivial. Where there are many possible graph-paths, a systematic method of route selection is used, described next.

To select a suitable route, edge weights are added to the input graph and a minimum weight spanning tree is calculated. In this tree representation only one graph-path exists between the start and endpoints, and which particular route is selected is influenced by the selection of edge weights. Weights are associated with edges based on three types of features:

- length l, the physical length of the edge in world coordinates;
- importance i, the combined vertex degree of the edge's end points; and,
- landmark proximity p, the number vertices between the edge and the nearest landmark.

The sum of these three properties is then calculated for each edge:

$$w(e) = \alpha l - \beta i - \gamma p$$

Modifying the multiplicative constants α, β, and γ therefore gives rise to a variety of possible routes: increasing α favours short edges; increasing β steers the route through highly connected vertices; and, increasing γ causes the route to pass close to, or through landmarks. The effect is to provide support for differing scenarios, for example the shortest route minimizes animation time whereas passing close to landmarks could aid the formation of survey knowledge.

4.3.2 Animated Pan and Zoom

Once the network path to be followed has been calculated, the next step is to determine the 3-D path that the camera must follow to provide an effective animated view. Tversky [128] conducted a thorough review of empirical evidence of the effectiveness of animation and found that to be understood, animations must adhere to two principles:

congruence between the changes over time and the conceptual information that is being conveyed; and,

apprehension of the content of the external representation.

An example of the application of these two principles to produce effective route maps of the real world has been reported before [2]. Specifically, long roads were shortened and the angle at junctions was increased on the basis that the junctions are important decision points whereas long roads give little useful information to support navigation. Furthermore, distorting the angles did not lead to difficulties since the exact angles are not comprehended.

An alternative method to illustrate changes over time is to provide multiple views of snapshots at key points, allowing the viewer to freely shift attention between the time-

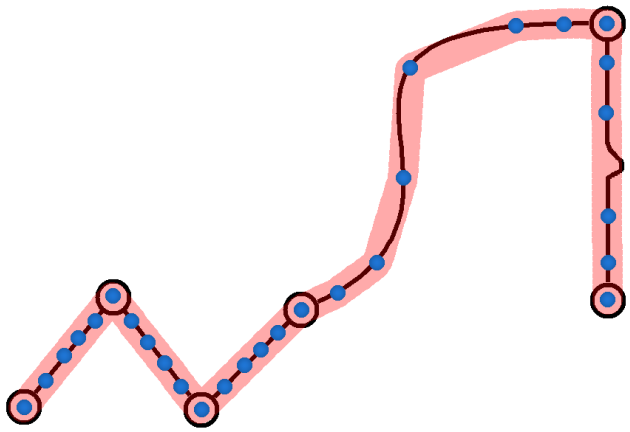

Figure 4.15: The camera path is formed by a sequence of ‘nodes’ placed along the network path. As the time to move the camera between nodes is fixed, and more nodes are placed close to network nodes, the effect is that important junctions are in view for more time than long edges. This method also ensures a ‘slow-in-slow-out’ effect at the start and end of the animation. Uninformative undulations in edges are also smoothed out.

steps that carry the most information. Instead, a continuous pan and zoom animation is used, with the benefit that tracking objects allows the viewer to maintain object-association between views. In CoronaScope the layout of the network is not distorted, instead the speed of the change in view is modulated, so that more time is available to perceive junctions while long edges are traversed relatively quickly and bends are smoothed out. The animated effect is controlled by adding a sequence of ‘nodes’ to the network path that each represent a camera position, then moving the camera in a linear interpolation between nodes (note that these nodes are not actually rendered). The amount of time allowed to move from one node to the next is fixed, with the result that a short distance between nodes appears slower than a long distance (see Figure 4.15). Nodes are arranged along the network path so that more time is available to perceive complex regions of the network with many turns, while long edges are traversed more quickly. The arrangement of nodes means that undulations in long edges that contribute little useful information about the network path are removed, though varying the zoom level (described next) means that network edges remain within view. Limiting the maximum distance between nodes on the animation path avoids transitions that are too fast to apprehend (as is the case for very long edges), and allows the path to approximately follow curved edges.

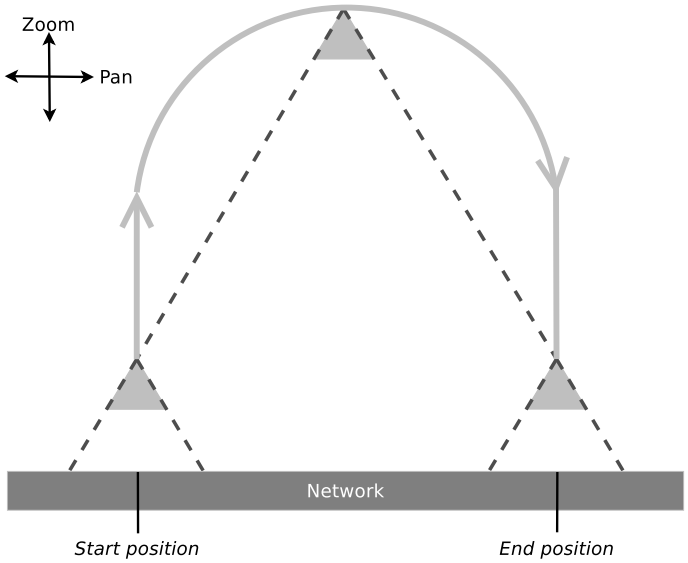

Figure 4.16: The zoom level during the animation is varied so that the at the mid-point of the animation both start and end points are in view, to enable local-global association. At the end of the animation the zoom level is restored to the original level.

Local-global association is the cognitive cost associated with making sense of local features within the structure of the global context. One option is to arrange the zoom level so that the entire network is in view, however doing so may mean that the path being followed becomes too small to be discerned as the edges it follows are lost in clutter. Instead we attempt to ensure the start and end points are made visible in one view to at least provide an overview of at the path being followed. Generalized formulae for calculating smooth camera paths through scale-space have previously been determined [50, 131], but both methods depend on metrics such as shortest-path and optical flow rather than the application-specific requirement here. As illustrated in Figure 4.16 the zoom level is varied such that at the mid-point of the animation, the entire path is within the view. Zoom level is restored at the end of the animation on the assumption that the level of detail at which the user was viewing the start feature is the same at which they wish to view the feature at the end of the animation.

An additional visual cue as to the network path being followed is created by highlighting the relevant edges with a light, semi-transparent poly-line. A dashed line drawn directly between the start and end points is used to encode the case where a graph-path was not found, the broken line suggesting a weaker link. Illustrated examples of following paths in a network using the animated pan and zoom tool can be found in Chapter 5.

4.3.3 Revisiting

The animated pan and zoom method as described here attempts to promote congruence and to facilitate the user in maintaining both object-association and local-global association. As Tversky noted [128], there are several challenges to apprehension that come in to effect when using animation, but interaction can provide the necessary tools to allow the user to overcome these difficulties. One problem is that key information can be missed if the animation moves too quickly, particularly when the view contains many edges. Conversely, an animation that is too slow can cause frustration amongst users [13]. Although the technique described above reduces overall animation time by moving quickly over uninteresting sections, a simple and effective addition is the provision of a speed control so that a user may adjust the animation according to their preference.

Missed information can also be obtained by *revisiting* points along the animation path, allowing the user to selectively view features of interest. The provision of specific support for revisiting previous view states reduces the "gulf of evaluation", and encourages users to explore the network [73]. Specific tasks that benefit from revisiting support include:

- comparison of two features;
- exploring digression points;
- returning to a known point to become reorientated in the network.

In CoronaScope a number of camera positions (nodes) are stored in a view history cache, Since each node is simply the x,y,z-position of the camera, a few hundred nodes can easily be stored. Following an animated pan and zoom the user can interactively move through the cached views by pressing the left/right arrow keys. Direct support for comparison by moving between any two features can be achieved by interactively annotating the two features, then using a combination of animated exploration and user-controlled revisiting along the one or more network paths that can be found by varying edge weights during route calculation.

4.4 Summary

The design space of off-screen navigation was surveyed as a potential solution to the problem that landmarks are not visible within the view of a multi-scale system such as pan and zoom. Existing solutions are based on topological navigation that limits movement to adjacent nodes in the network, or suffer from distortions and cluttering in the corners of the view. To counter this I contributed “CoronaScope”, a novel off-screen visualization that avoids the problems described by fixing proxies to a circular bezel in the centre of the view, thereby providing an on-screen representation of the overall configuration of landmarks.

To deal with the visual clutter created by overlapping proxies, an efficient algorithm was devised that removed overlaps, maintaining the original order of proxies and distorting each by the smallest amount possible. In this way the relative arrangement between off-screen landmarks could be comprehended. In this case, encoding the absolute distances between landmarks was subject to a trade-off against a simple, easily perceived view.

The framework created by the landmark overlay, the bezel, and the proxies provided the basis for specific interactive tools to directly support path-following. Path-following is a fundamental task in networks, and most studies and tools are concerned with incremental navigation between adjacent nodes. To enable following longer paths I proposed the use of automatic pan and zoom, tailored to following the edges of a network through scale-space. The addition of a history tool enabled views to be revisited, a task that often forms part of network exploration.

In all cases, the design decisions taken during construction of the new designs were grounded within perceptual evidence and cognitive theory, with the aim of providing navigational information in a way that is readily perceived, and structured in a way that is congruent with the user’s internal representation of the global network space.

Chapter 5

Analysis: Three Case Studies

The design of navigation aids for networks based on theories of navigation in related environments, has been described in previous chapters. In this chapter three example data sets are presented with the aim of demonstrating the selection of landmarks, and the potential benefits of carrying out network exploration with landmarks overlaid upon the network representation. It is shown that the initial selection of landmarks is key to the utility of the overlay, but that even in extreme cases of clutter, the new tools offer some analytical support.

5.1 Case Study One: A Map of the Heap

This example demonstrates the selection of landmarks in a graph that represents the structure of objects in heap memory storage of a running C++ program. The example was first used in [91], and has been reworked for this thesis. Each vertex represents an instance of C++ class or basic data type (integer, float, etc.). A directed edge is formed when an object instantiates another object in heap memory. The data was extracted by taking a snapshot of the objects allocated on the heap from a running VTK program. The program contains a visualization pipeline that reads graph data in from a file and displays a network.

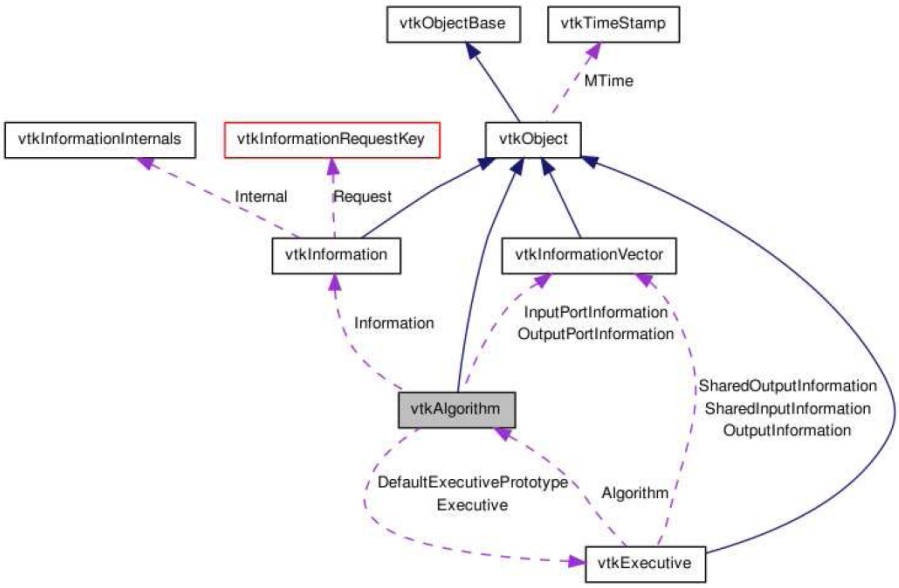

Figure 5.1: The collaboration graph of a vtkAlgorithm taken from Doxygen documentation. The blue edges represent inheritance relationships, while the pink edges represent 'has-a' relationships. The 'has-a' relationships of several key objects are used as the basis for the set of landmarks in a heap memory graph.

5.1.1 Landmark Selection

Documentation and training material was used as a source of insight into the concepts and abstractions that VTK users will be familiar with. In introducing programmers to the architecture, VTK documentation such as [109] describes two major object models:

- The *visualization model* refers to the part of the system devoted to converting raw data to a geometric representation which is then rendered by the graphics model. Two key classes underpin implementation of the visualization pipeline: vtkDataObject serves as the base class for a family of specialized data types that are passed along the pipeline; vtkAlgorithm likewise provides a common base type for the filters that operate on the data objects at each stage in the pipeline. These two classes are fundamental in pipeline execution and thus meet the *significance* criterion for landmark selection.

- The *graphics model* covers those classes used to take data from a pipeline and assemble a graphical scene. Examples include vtkActor, vtkLight, vtkCamera and vtkTransform. A specific pipeline will contain concrete instances of subclasses that are appropriate to the underlying graphics library.

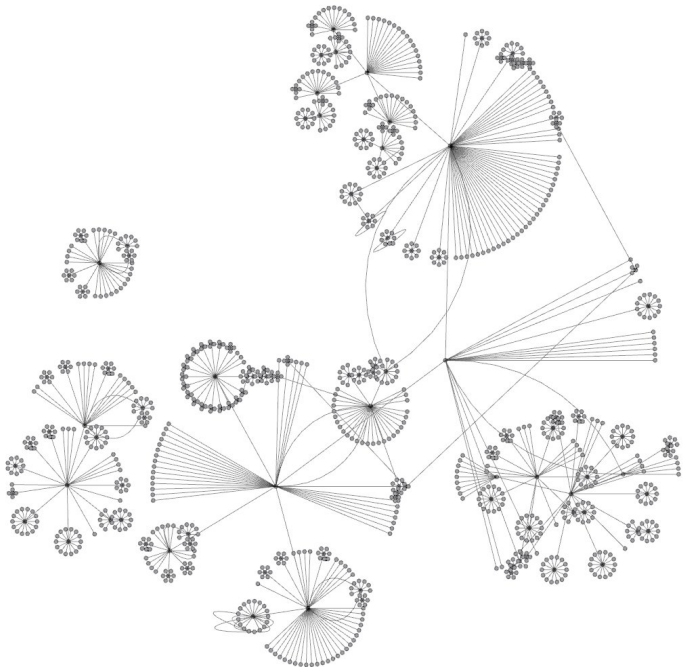

Figure 5.2: VTK pipeline heap memory network without explicit landmarks.

5.1.2 Analysis

Instances of the above classes (and their subclasses) form the backbone of a VTK pipeline. They are part of the user's vocabulary and are explicitly instantiated by the user in their own code. It is likely therefore that these classes are significant in linking regions of the heap graph to components of the user's code. However, in examining the heap, individual class instances are unlikely to be distinctive or perceptually helpful. Instead, a landmark is defined to be the *collaboration graph* of such a class. In the case of VTK, this graph is obtained from the online documentation which is generated automatically from the source code with the Doxygen tool. Each landmark therefore represents a functional unit consisting of several C++ objects.

The example in Figure 5.2 shows the resulting network with no landmarks highlighted, with node and edge labels removed for clarity. In this case the graph is a special case known as a quasi-tree, a tree with a very small number of additional edges. Having this property means that the graph can be embedded using a tree layout algorithm, with the additional non-tree edges being added after the tree layout step. With no explicit landmarks the network can be seen to comprise three connected components, each of which is a VTK filter (the pipeline connections between filters are not included in the graph). In this case: a delimited text file reader that outputs a vtkTable; a filter that converts

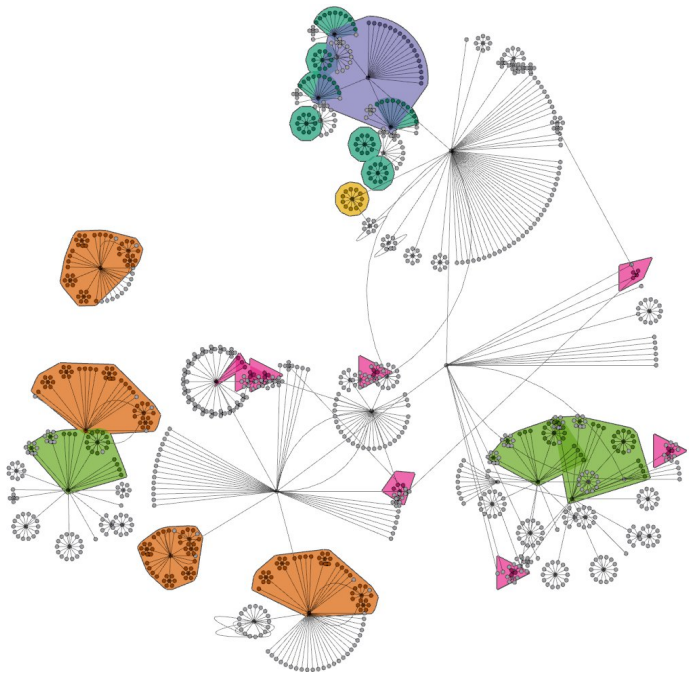

Figure 5.3: VTK pipeline heap memory network overlaid with a selection explicit landmarks. The landmarks give high-level structure to the network, highlighting familiar sub-structures within the view.

from the vtkTable to a vtkGraph data object; and, a complex filter that performs layout, geometric encoding and contains the graphics/rendering model. Upon closer reading an unusual looking cluster of nodes can be seen near the centre of the network which turns out to be the command/observer structure used to propagate interaction events.

The addition of explicit landmarks (Figure 5.3) immediately provides information to the viewer about the locations of some key, well understood and expected components in the heap memory graph. At the core of each filter is a vtkAlgorithm object (orange), and there are three vtkDataObject types (green). At the top of the network is a vtkCamera (purple), with a vtkLight (yellow), and various associated vtkTransforms (blue). Distributed throughout are a number of command/observer structures (pink). Furthermore, the display of landmarks as a semi-transparent overlay forms a separable visual level, so that the nodes and edges of the network are not obscured.

The use of colours that are easily named promotes remembering [136]. Given sufficient exposure to the landmarks view, the connection between a given colour and the referred to class is soon learned. At this stage a brief glance at the display is often enough to recognize and locate the significant features of interest. From the point of view of heap-memory debugging, the presence or absence of these major features can provide clues to

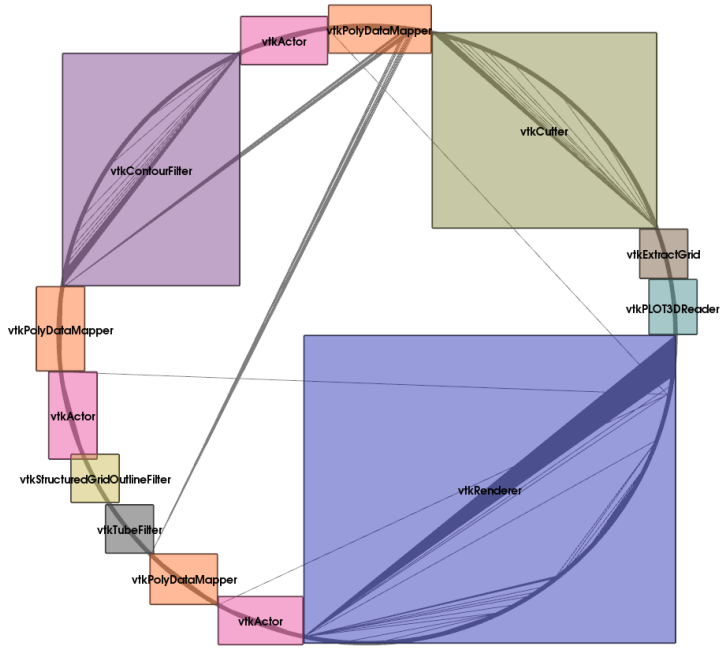

Figure 5.4: An overview of a more complex VTK pipeline heap memory graph. The circular layout is not based on semantic or topological input, therefore creating an arbitrary arrangement of nodes. The addition of landmarks exactly reflect the objects (VTK filters) that a programmer would have directly instantiated, thereby creating an immediately recognizable layer of abstraction, above the specific details of nodes and edges.

solve problems. For example, the small connected component contains a vtkAlgorithm but no vtkDataObject is present, as would be expected.

The tree-like data used for this example meant that the layout algorithm was able to capture and present the semantics of the data to a large extent, suggesting that the landmarks may not provide much further assistance beyond some high-level overviews. One positive effect noticed by the author, is that the simpler, abstracted view is easier to remember, and one can continue to reason about the data long after actually viewing the network. Using a more complex example with a circular layout, the landmarks provide a complete overview of the objects instantiated in heap memory (Figure 5.4). In general, a circular layout is simple to compute but takes no account of the topology, or semantics of the network. The radial layout of nodes in itself removes any ordering or hierarchy effects, desirable or otherwise. In this case, the landmarks are able to solely provide the semantics of the network.

5.2 Case Study Two: InfoVis Co-Authorship Network

This example data set was used in the IEEE Information Visualization (InfoVis) contest [46] in 2004, and has since become a benchmark data set having been used to demonstrate graph visualization techniques in several publications since then. The data set contains complete details of all the papers published at InfoVis during the years 1995-2002. In this example a graph was generated from the data by linking all authors who have worked together on an article, referred to as a co-authorship network, so that vertices represent individual authors, and edges are instances of co-authorship for a particular paper. The original graph contained one large connected component plus hundreds of small connected components containing only two or three authors. For this analysis, and for the purpose of illustration, the many small components were filtered out to leave just one large connected graph.

5.2.1 Landmark Selection

In social networks such as the one used in this case study, *betweenness centrality* is often used to determine the importance or influence of particular individuals in a network [17]. Betweenness centrality is a graph-theoretic metric based on the number of times the shortest path between every pair of vertices passes through a given node: the more such paths that pass through a vertex, the higher the degree centrality value. To select landmarks for the co-authorship network betweenness centrality was calculated and the most significant ten vertices were chosen. This appeared to be a reasonable measure with the influence of MacKinlay, Card, and Robertson clearly represented, as well as other well known contributors.

5.2.2 Analysis

As the landmarks represent well-recognized names in the field of information visualization their immediate effect is to label the clusters of nodes (see Figure 5.5). The clusters represent close collaborations, presumably between researchers from the same institution (supervisors and students for example), with the landmark node calling out the senior member of the group. The benefit is that the viewer can quickly become orientated, and the otherwise arbitrary arrangement of clusters is easily linked with the semantic content.

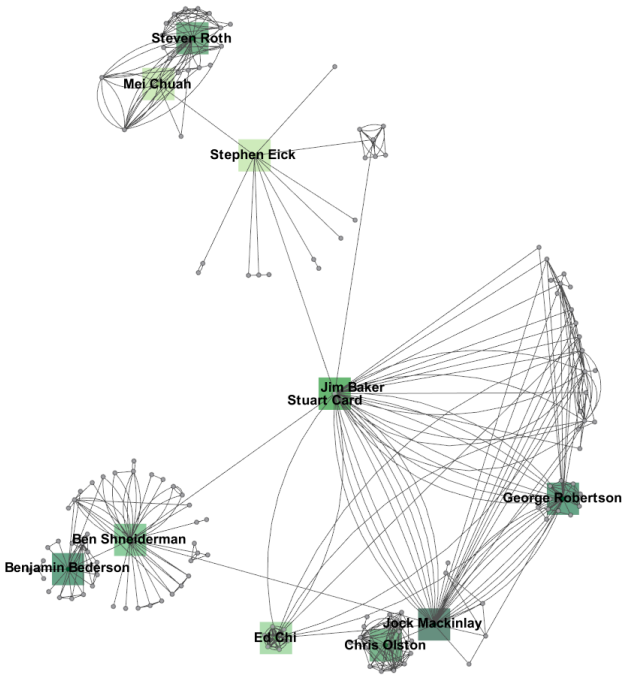

Figure 5.5: Landmarks as an overview.

In addition to an *overview* of the network, CoronaScope can also present *details-on-demand*. Consider a scenario where the user is zoomed in to the cluster at the top of the network containing Steven Roth, and wishes to answer the question, "what, if any, is the relationship between Steven Roth and Ben Shneiderman?" This conceptual question translates to a network task of path-following, to determine if there is a graph-path between the two nodes. From the current view, the user can select the off-screen proxy that represents Ben Shneiderman, and observe the animated pan and zoom that follows. The three panels in Figure 5.6 illustrate the main stages of the animation, beginning with Steven Roth, zooming out to reveal the entire path in the global context of the network, and coming to rest at Ben Shneiderman, zoomed in so that the local context is discernible. At this point the history function allows the viewer to rapidly return to any position along the route, perhaps to inspect the local context around the group of landmarks that was passed along the way.

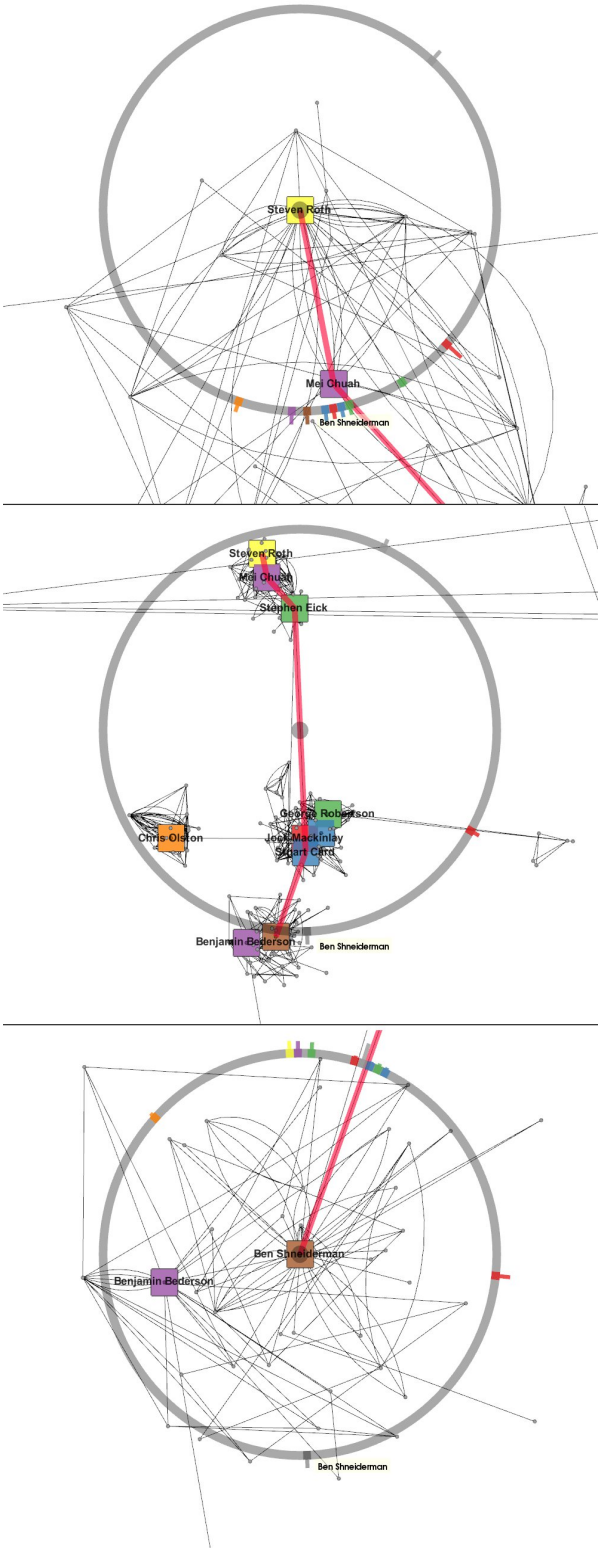

Figure 5.6: CoronaScope provides direct support for following network paths through scale-space. An animated pan and zoom is initiated by selecting a proxy, the network path is highlighted, and an animated camera movement moves the view to focus on the selected off-screen target.

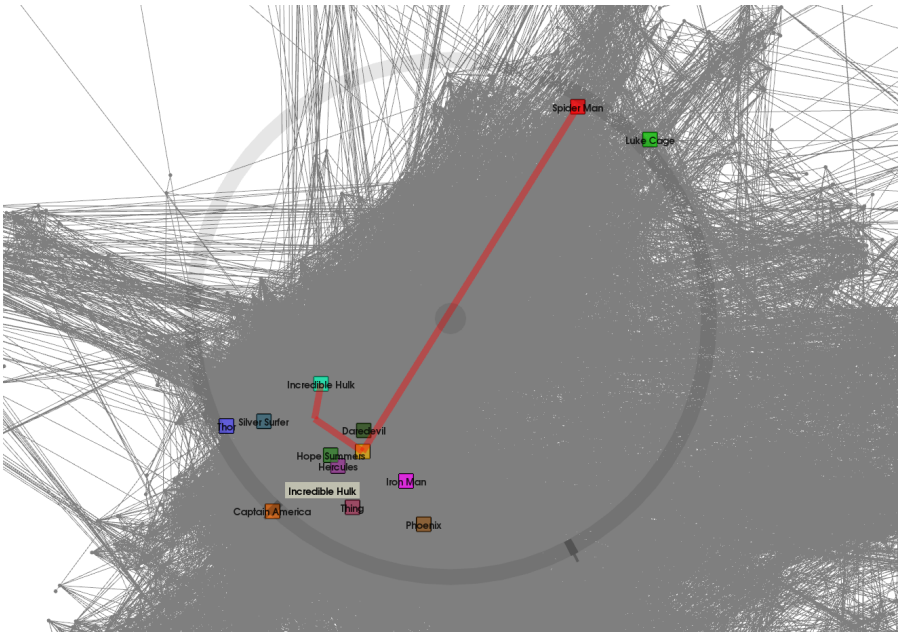

Figure 5.7: Path-following in an extremely dense network. Despite the heavy cluttering of edges, the animated pan and zoom tool can be used to identify the relatively simple paths between familiar objects.

5.3 Case Study Three: Marvel Comics Network

The data set for this example is from a paper that studied which Marvel comics super-heroes have appeared together in at least one episode. The study found that the network has some graph-theoretic properties in common with a real social network [4]. It is included as an example here due to the relatively large size and high density, having over $6,000$ vertices (super-heroes) and approximately $40,000$ edges (co-appearances).

5.3.1 Landmark Selection

The top ten most popular super-heroes were determined via a web based poll in which around $7,000$ people voted which were then used as landmarks. Again this method resulted in the appearance of some highly recognizable names in the list of landmarks, for example Captain America, Spiderman, and Incredible Hulk.

5.3.2 Analysis

Unlike the other examples shown the overview gives very little information: clutter caused by overlapping edges almost completely removes any relational information, and the landmarks are grouped within one area of the network. This could be interpreted as a complete failure, though the grouping of landmarks suggests an opportunity to filter the network to focus on the small region occupied by the most significant actors. In the case of following paths between vertices the animated pan and zoom technique proves to be useful. Despite the large number of over-plotted edges, highlighting of the path being followed reveals that the structure between landmarks is relatively simple (see Figure 5.7).

5.4 Summary

In this chapter three case studies were set out with the aim of demonstrating methods of landmark selection, and how the resulting overlays can benefit network navigation. Initially the landmark overlay provides a high-level, semantic overview of a network, and with repeated exposure one can expect the user to become familiar with, and is more likely to remember, the overall structure of the network. Direct interactive support for the specific tasks of following long paths and revisiting was demonstrated, although a reasonable distribution of landmarks across the network is required to provide comprehensive coverage. In the final chapter these findings are reviewed and recommendations for further work are made.

Chapter 6

Implementation

A prototype graph visualization system known as “CoronaScope” was implemented to experiment with the network navigation overlay designs described in earlier chapters. In this chapter I justify the choice of the Visualization Tool Kit (VTK) as an application framework, and describe details of the implementation of CoronaScope. Since VTK uses a *data-flow pipeline model*, and was originally designed to support scientific visualization, its use for information visualization is somewhat experimental. While the modular design enabled the logical separation of network representation from the implementation of the new navigation tools, the assumptions implicit in the pipeline model as intended for scientific visualization led to some challenges. Following an introduction to the data-flow pipeline model and graphics models used by VTK, I report on my experience and describe the design of new components that, combined with existing features, enabled implementation of the CoronaScope application.

6.1 Rationale for using VTK

Figure 6.1 shows a series of conceptual layers through which a visualization application communicates with the underlying graphics hardware. The role of a visualization library is to provide developers with an environment that is organized around visualization concepts, and to translate those concepts into the language of the underlying graphics library, freeing the developer from low-level concerns. Systems supporting graph visualization span a range of software architectures. At one extreme are language extensions and graph libraries such as BGL [115] and OGDF [25], which provide suitable algorithms

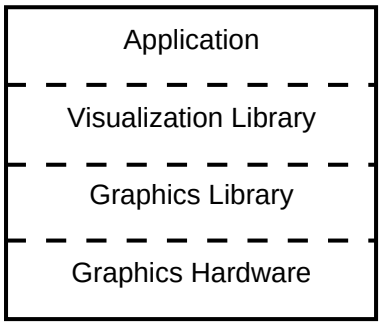

Figure 6.1: A layered systems architecture for information visualization development. A visualization library provides functionality in the language of visualization and interfaces with the native graphics library/hardware. This separation allows concerns of graphics generation to be abstracted away from those of visualization development.

and data structures, but lack support for graphics and interaction. At the other extreme are specialized, monolithic tools, such as GraphViz [52] for producing static drawings, or systems that provide a suite of network-specific interaction techniques, for example CGV [123]. The space in between is occupied by extensible systems and modular tools, some of which (e.g. Tulip [6] and Gephi [11]) are primarily for graph visualization, while others, including Prefuse [61] and VTK [109], are more general visualization tools with components for graph visualization.

A primary contribution of this thesis is the realization of a modular navigation layer, separated from concerns of graph representation. VTK [109] is an extensive open-source software library aimed originally at scientific visualization and recently extended to include components for information visualization [139]. VTK is based on a demand-driven pipeline architecture [57], where each component in the pipeline performs some transform on its input data before passing the data along to the next component in the pipeline. The output of the pipeline is a set of geometry that is passed onto a graphics model that prepares the final scene for rendering by graphics hardware, and provides facilities for interaction. Requests for new or updated data are passed upstream to the component that can fulfil the request. The benefit of VTK's modular architecture for the work in this thesis is that an existing network visualization pipeline can be enhanced by adding a new pipeline branch, taken from the graph layout filter, and into the new navigation components. The output of this new branch is then added to the renderer along with the network representation. The original pipeline remains unchanged and distinct from the additional navigation support.

From a visualization users point of view, the advantage of the modular architecture is

that it provides the flexibility to create custom pipelines to support particular combinations of task and data with relatively little programming knowledge. The user must have some understanding of the visualization process, though pipeline building tools such as ParaView [119] provide a graphical user interface that can assist with design. This advantage also holds for visualization developers, as only specific modules need to be developed to create new designs as many existing data structures, graphical techniques, and interaction components are already available in the tool kit. One further advantage of VTK is that it is somewhat mature, and has an active community of users and developers. As an open-source product, VTK provides an online review system to allow code contributions from the community to be subjected to peer review, before being added to the publicly released code. Two of the newly implemented components produced to build CoronaScope were submitted for review and have been included in the public release of VTK since version 5.8, namely vtkConvexHull2D and vtkGraphAnnotationLayersFilter described in the second part of this chapter.

Having justified the choice of framework, the following section contains a detailed description of a basic graph visualization pipeline, and the new navigation components. VTK provides bindings for Java, Python, and Tcl, but all code for this project was written in the native C++ language as doing so results in the fastest possible executables.

6.2 The Basic Network Pipeline

A minimal network visualization pipeline can be constructed from four main components, approximating the data states found in the information visualization reference model [21]:

- a data source (raw data and data tables);
- a layout algorithm (visual structures);
- a representation step that maps topological structure and vertex positions into geometric entities (visual structures); and,
- mapping the representation from geometric entities to graphics primitives (view).

However, before considering the specific example of a network pipeline, it is necessary to describe the pipeline execution model of VTK.

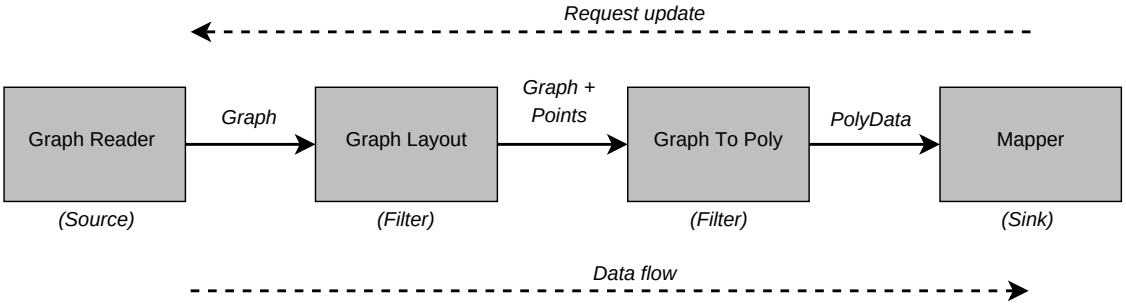

Figure 6.2: A basic network visualization pipeline in VTK. Each *filter* produces an output *data object* that is passed as input to the next filter in the pipeline. Execution of the pipeline is triggered by the *sink* filter which asks for updated data from its upstream filter in the pipeline. Update requests pass along the pipeline until each filter is up to date and data flows back to the sink. Update requests only propagate as far as is needed: since each filter can store a copy of its data object, only filters whose parameters or input data have been modified need to re-execute.

In VTK terminology each pipeline component is referred to as a *filter*, and *data objects* are passed from filter to filter, so that data flows along the pipeline. A *source* is a filter that has no input, rather its output data object is constructed parametrically in the case of simple geometrical objects, or as a result of reading data from a file or network stream. The visualization pipeline is terminated by a *sink*. The sink is usually a type of mapper that converts the geometric data from the pipeline into a set of graphics primitives suitable for display by the graphics model, described in the next section. A filter contain two major components: an *algorithm*; and an *executive*. The algorithm component is responsible for performing some computation that results in some output data object. The executive is part of the pipeline execution system that is able to determine whether its output data is up to date, and if not, to run the algorithm to generate the updated data. Figure 6.2 is an overview of the pipeline execution process, showing the direction of update requests and the direction of data flow.

Pipeline execution is triggered by a *sink* object, typically in response to a request from the graphics model to render the scene. The sink object passes the request to its input filter, which checks to see if its data object is up to date. If so, execution returns to the sink object, otherwise the algorithm must compute the data, which may require a request for data from its input and so on. Update requests are propagated back down the pipeline until the required data is found, possibly as a *source* algorithm. By default each algorithm's output data is cached so that it need only be recomputed when input data, or a parameter of the algorithm is modified. This mode increases memory requirements in exchange for improved pipeline update performance. Note that reference counting

reduces the need to copy data objects, so in the network pipeline described here, the graph with layout is simply a reference to the initial graph data object with an additional set of points. Several alternative pipeline execution models are available, for example using multiple-passes to stream subsets of data, or distributing pipelines across multiple processors but these are not used here.

A basic network visualization pipeline then, begins with some source data streamed into the pipeline by a file reader. For this project the Tulip file format [6] was used as VTK already provided a suitable reader. The output of the reader is a vtkGraph data object which at this stage contains only a description of the topology (vertices and edges). vtkGraphLayout is a filter that takes a vtkGraph and calculates a layout of the vertices according to the chosen vtkGraphLayoutStrategy. Strategies are instances of layout algorithms, of which VTK provides a small number of basic methods such as circular, force-directed, and clustering. To enhance the set of default layout algorithms, I created an adaptor to the Open Graph Drawing Framework (OGDF) which includes more modern algorithms, in particular the fast multi-pole multilevel (FM^3) type algorithms [9]. The layout step imbues the vtkGraph with a point (x,y-coordinate) for each vertex. By default the edges are drawn as straight lines between their end points. Edge layout may be carried out as a separate step, adding additional points to each edge to create arcs, or bundles.

All data objects in VTK ultimately derive from the same base class. The input and outputs of filters are strongly typed so that only those specific data types a particular algorithm is designed to use may be connected to a filter, otherwise an error message is issued and pipeline execution terminates. Data objects that already contain geometry (in VTK these are known specifically as datasets) can be mapped directly into the graphics pipeline, as is often the case with scientific visualization data formats, a structured mesh for example. Otherwise, it is necessary to use a filter that is able to convert the abstract data into a collection of polygons, lines, and other graphics primitives. The class vtkGraphToPolyData is a filter that takes a vtkGraph as input and generates a glyph at each vertex position, and lines or poly-lines for the edges. Finally, a mapper converts the polygon data to a format specific to the underlying graphics platform, and the network can then be represented within the graphics model, described in the next section. Execution of the pipeline is triggered by a request for it to be rendered. The generated data remains in memory until either a new file is requested, or a new layout algorithm is selected by the user, via CoronaScope's graphical user interface (see Figure 6.7 for screen shots of the user interface).

6.2.1 Graphics Model

The graphics model in VTK is an abstraction layer that separates the underlying platform-specific graphics application programming interface (API) from the concerns of visualization data. This decision is largely historical, since when VTK was conceived there were several competing graphics formats. Most objects within the graphics model are instantiated by static factory classes that transparently provide concrete instances of the classes that support the underlying library. For now at least, OpenGL is the predominant platform used in scientific and information visualization, however the layer of abstraction has allowed advances in hardware design and GPU programming to be incorporated into VTK without modifying the high-level visualization pipeline.

VTK uses a film production metaphor to describe the rendering API. *Actors* represent visual objects arranged in a scene with *lights*, and a *camera* to represent point of view from which the scene is rendered. The geometry of an actor is provided by a *mapper* as described in the previous section, and the actor itself contains positioning, and possibly colour and texture information. For CoronaScope, an orthographic view on to a 2-D plane was used, so that the camera remains above the plane, pointing down the z-axis. Pan and zoom controls enabled the user to interactively move the camera in the x, y (pan) and z (zoom) directions only. Lighting for the scene was similarly fixed, using the single default global light source. A vtkRenderer is responsible for managing the rendering of all its actors, and a vtkRenderWindow represents a rectangular window or viewport in a graphical user interface and can contain one or more renderers. It is this combination that presents that final view of the network, including removal of hidden objects by frustum culling and rasterization.

Closely related to the render window is a vtkRenderWindowInteractor. This class provides interaction support by routing mouse, keyboard, window, and timer events into VTK's event handling mechanism. Event handling in VTK is based on the command/observer design pattern, that defines a one-to-many dependency between objects so that when an interaction event occurs, all registered observers are notified automatically. This allows multiple observers to be added to the interactor at run-time, after which observers respond to the events by running their associated command method. A typical example is the routing of mouse events to modify the camera position, producing pan and zoom interactions. The observer mechanism is organized on a priority basis, and further processing of events can be prevented so that, for example, mouse clicks can be intercepted by an overlay layer and not passed on to the underlying plane. Various picking opera-

tions are possible, with hardware-based methods being fastest but providing only pixel coordinates, whereas slower software methods such as ray casting, or the use of space-dividing data structures, can be used to identify specific actors or pieces of geometry.

6.3 The CoronaScope Application

Having described the VTK pipeline and graphics models, and illustrated a generic network visualization pipeline, in this section I describe the development of the CoronaScope software, a network visualization application that provides basic network visualization tools, plus the new components that provide custom support for network navigation. The navigation components are additional pipelines, that do not affect the existing graph visualization pipeline. Figure 6.3 shows how the new components are branched from the graph layout step. Also shown is the correspondence with the information visualization reference model [21], first introduced in Chapter 2.

6.3.1 Landmarks Overlay

Landmark definitions are represented using vtkAnnotationLayers, an existing VTK data object type that stores layers of selections of vertices. Each selection is a set of vertex indices which can be used to access the x, y-coordinate of the corresponding vertex from the graph layout data. This creates an hierarchy: a set of layers; layers of landmarks; individual landmarks; and, vertices. There are no restrictions on the membership of a landmark so a vertex may be contained in more than one landmark. Similarly landmarks can be fully contained within landmarks to (potentially) create an hierarchy of landmarks. Additional indexed arrays store text labels, a representative vertex, and colour information.

Landmarks can be streamed in to the pipeline using an XML file reader, or added interactively using the mouse to make a selection of vertices in the view, achieved by observing selection events generated by the render window. To provide perceptual support for making selections, a “rubber-band” bounding area is drawn directly in to the renderer’s pixel buffer. When a selection is made by the user, the set of vertices is added to the vtkAnnotationLayers data as a new landmark. Two new filters were developed specifically for this component of the project, described in the following paragraphs, and illustrated in Figure 6.4.

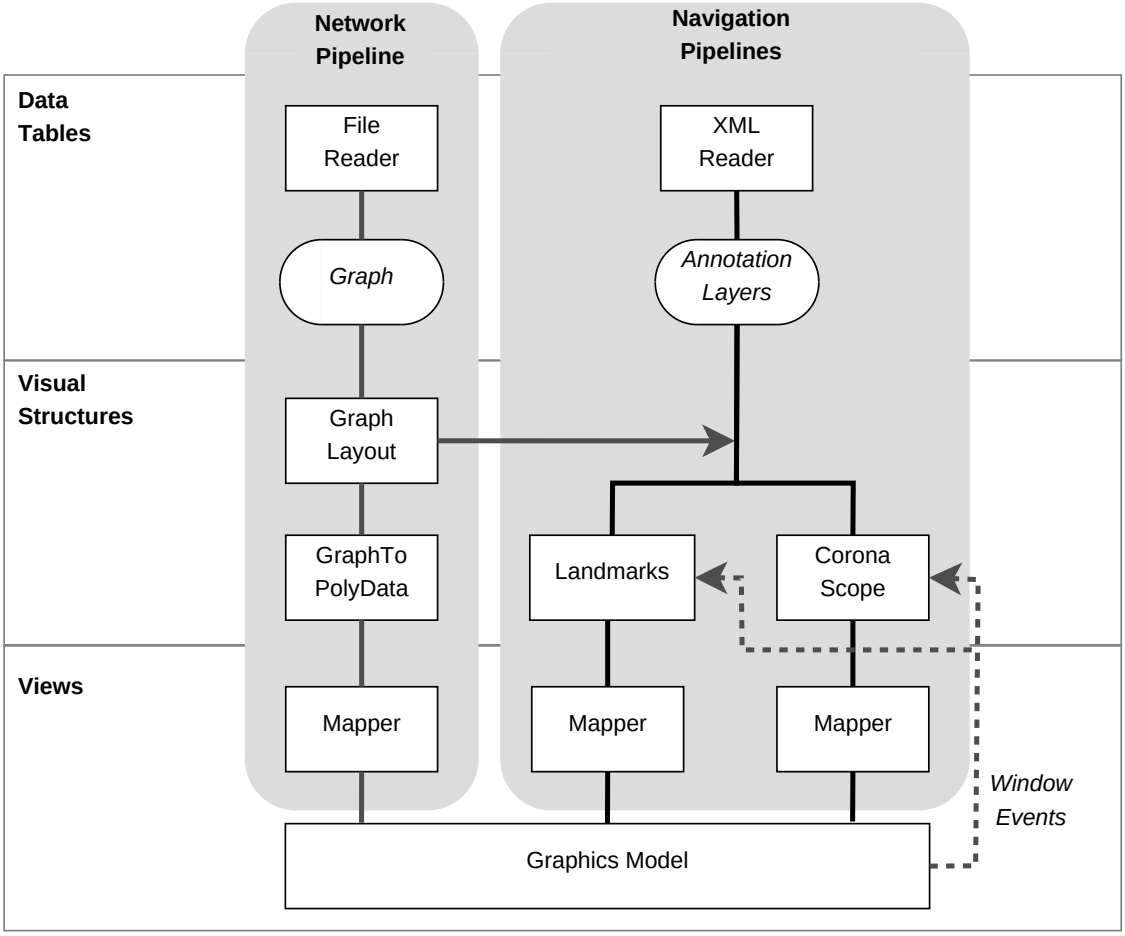

Figure 6.3: Implementation model showing distinct pipelines for network representation and navigation: the addition of the navigation tools leaves the network visualization pipeline unchanged. A branch is taken from the graph layout module which, along with landmark definitions (sets of graph vertices) in the vtkAnnotationLayers data, form the input for drawing the landmark and off-screen overlay representations. Pan and zoom causes window events that change the scale and extent of the view, so these events are passed back in to the navigation pipeline to provide for landmark scaling to guarantee a minimum pixel size, and to enable off-screen landmarks to be determined.

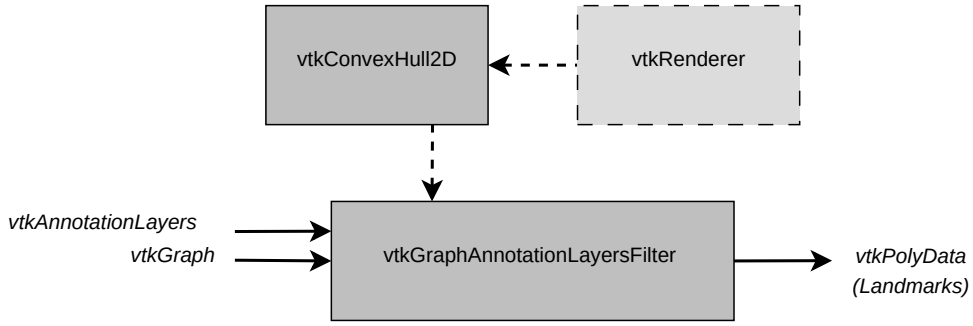

Figure 6.4: The constituent points of each landmark are processed by vtkConvexHull2D to determine the convex hulls to draw. Information from the renderer is used to determine the current camera position so that landmarks are scaled to satisfy the requirement for minimum pixel dimensions.

vtkConvexHull2D generates a polygon that is the convex hull of a set of input points. A second input from the renderer provides the current camera position, so that minimum polygon dimensions can be calculated in pixels, ensuring that the landmark is a constant size regardless of the current scale (zoom level). The filter is explicitly invalidated in response to camera-move events with the result that the landmarks are recalculated the next time an update request is received, typically the next render frame.

vtkGraphAnnotationLayersFilter takes the data from vtkAnnotationLayers plus the output of the graph layout filter to draw polygons that form the convex hull of each landmark. The filter takes the vertex indices from each landmark in the vtkAnnotationLayers and looks up their points from the graph layout data. This set of points is then passed to the convex hull routine described in the previous paragraph. The output of this filter is polygon data of the set of convex hulls that represent the landmarks which finally, is added to the renderer along with the network representation.

Since both the new filters must respond to events generated in response to camera movements, any delay in re-execution of the pipeline could cause an interruption to rendering, leading to a reduction in responsiveness. In the pipeline model, any one filter that needs to be updated causes all the filters between it and the sink to also re-execute. For this reason it was beneficial to ensure that these particular filters were not only very efficient, but also positioned close to their sink, minimizing the time required to bring the pipeline up to date. In this way, coupled with the relatively small number of landmarks, delays in rendering are avoided.

6.3.2 CoronaScope Overlay Widget

The CoronaScope overlay refers to the navigation component proposed in Chapter 4 that provides a representation of off-screen landmarks within the view. While window events can be used to trigger changes of view point to produce pan and zoom effects, users may also wish to interact with the scene in more complex ways. For this purpose VTK provides a "widget" API that allows an interactive mechanism to be supported by a visual representation so the user can control the operation, for example manipulating a bounding box around a 3-D volume rendering. The widget API is implemented in two parts that separate interaction and changes to the scene from the graphical representation of the widget. In Figure 6.5 for example, the vtkFlightMap, responsible for animating the camera is part of interaction, while the vtkDiskSource/vtkSectorSource filters provide the graphical objects used to create proxies and the bezel.

vtkCoronaScopeWidget was developed as a widget that is rendered into the *graphics overlay plane*. The overlay plane is defined in display coordinates and always appears above the main scene. The overlay plane can be thought of as being attached to the camera so that changes to the camera position do not affect the appearance of the overlay, in this case the bezel and centre mark of the CoronaScope overlay remain fixed in position. Calculating and drawing the proxies is more complicated, since they must respond to changes in the view. This is achieved by observing pan and zoom events, which triggers the calculation of which landmarks are currently off-screen and their distance: these values are used to determine the position of proxies and length of the distance indicators. vtkDiskSource and vtkSectorSource are filters that generate basic geometrical shapes according to given parameters. Each proxy is created by appending two sectors and the set of proxies forms one polygon data output. The bezel and centre-mark are formed from disks, and combined in a second output.

The CoronaScope design requires that hovering over a proxy causes that proxy to be highlighted and its text label to be displayed. VTK provides a number of picking methods and which to use is a trade-off between speed and the level of abstraction at which the picked object is identified:

- hardware picking is fastest and returns only the pixel coordinate from the window;
- ray casting is an approximate method carried out in software that can return a reference to a specific vtkActor; and,

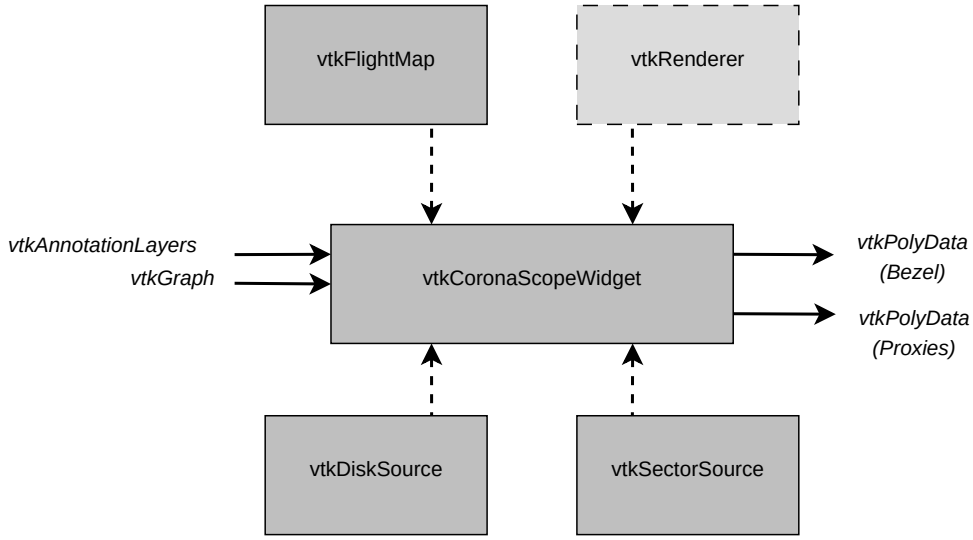

Figure 6.5: The CoronaScope widget generates the bezel and centre marks from two disk sources. The presence and location of each proxy is determined from the locations of landmarks and the extent of the currently rendered view, then each proxy item is formed by combining two sectors.

- space-dividing data structures take longer to compute but can accurately pick individual primitives in geometric data.

Note that the entire CoronaScope widget is represented by a single actor and each proxy contains many points and cells. This turned out to be a problem, as none of these methods can return item-level information about which proxy was picked, that is, the index into the vtkAnnotationLayers data structure: the information needed is lost when the data is encoded as geometry. A simple solution in this case was to associate an array containing text labels with the points and polygons when the proxies are encoded, effectively adding a large amount of redundant data to ensure its availability later in the pipeline.

Highlighting the selected proxy was a greater challenge since again, the item-level information about the proxy itself is not available. In this case, given a picked cell index, and hence the index of the landmark it referred to, it was then necessary to identify all the cells in the geometry data of the proxies with the same landmark index. Having identified all the geometry which belongs to the selected proxy, the obvious choice of temporarily changing the colour associated with it would require the underlying landmark data to be changed, and consequently would cause that section of the visualization pipeline to be recalculated. The approach taken instead was to draw a second semi-transparent proxy shape over the selected proxy so that when combined the effect is to increase the intensity of the colour. Clearly none of these approaches are scalable due to the linear

searches involved and the computational expense of picking geometric primitives.

The problems of obtaining item-level information about the underlying data from the geometrical data in later stages of the visualization pipeline seem to be a limitation of the VTK model. Since scientific visualization is generally concerned with continuous spaces that are interpolated, there is often no connection between a particular location in the view and a specific value from the underlying data set. At the time of writing, a new information visualization API is being created for VTK that is similar to a scene graph, with lightweight items that represent individual 'visual data items'. This new API may well have been a better choice for the highly responsive interaction required in the CoronaScope application.

vtkFlightMapFilter is a graph algorithm that adds edge weights to the input graph data. CoronaScope uses this when the user has selected a proxy to determine a route along which to move the camera, as described in Chapter 4. Once a route has been selected the path is first smoothed using Kochanek splines to avoid jerky camera movements. The animation is achieved using a timer callback that each time it is called, moves the camera to the next position. The path is stored so that the user can revisit the route using arrow key presses.

6.3.3 Desktop User Interface Integration

Qt is a cross-platform desktop application development framework with an extensive graphical user interface (GUI) module [34]. Coupling Qt with VTK allows desktop-style widgets to be used to display and edit data, while benefiting from VTK's visualization capabilities. A class is provided that wraps a vtkRenderWindow in a Qt widget for display, and synchronizes event mechanisms between the two frameworks. A key benefit of this approach is the ability to change parameters on VTK filters, or even reconfigure the pipeline during run-time. For example, a combo-box lists each of the available graph layout algorithms. When a new layout is selected by the user, an event is fired that ultimately causes the current graph layout strategy to be replaced with the new choice. Doing so invalidates the data flow in the pipeline so that the next time rendering triggers an update request, the graph layout will be recalculated using the new filter.

In the case of the list of layout algorithms described in the previous example, being a small amount of static data it is a simple matter to hard-code the possible choices.

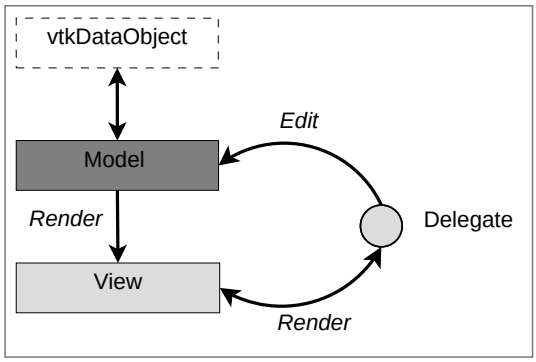

Figure 6.6: The model/view architecture as implemented in Qt provides a clear separation of concerns between managing data and updating the display. A model communicates with a data source and presents a standard interface to components that provide a view. The view obtains data via the model, and individual data items are rendered by a delegate object. Changes to the model are signalled to all registered views.

To maintain a table-view of current landmarks is more involved, since the data can be changed in several places: loading a new XML file via the user interface; editing the values using the table-view in Qt itself; and, interactively by making a vertex selection within VTK. For this situation Qt provides a model/view architecture to separate the data model from the view (see Figure 6.6). The benefit of this architecture is that changes to the underlying VTK data object are automatically reflected in the data model and any attached view is automatically updated to display the new values. Similarly, changes to the model by editing values in the table view are passed through to the underlying data object, causing the navigation pipelines to be updated. Figure 6.7 contains screen shots of the table of landmarks, and other GUI components used to control CoronaScope.

6.4 Summary

The rationale for choosing the Visualization Tool Kit as an information visualization development framework for this thesis was that it allowed the rapid construction of a generic network visualization pipeline that could be branched, with the advantage that additional navigation support could be provided as a distinct pipeline. From the user's point of view, such tools can be easily incorporated in their own visualization designs by composing pipelines from the required components. I described the implementation of several new VTK filters, and those that were used to create the landmarks overlay have since been incorporated in the public release of the VTK library.

Some of the assumptions implicit in the pipeline design, as it was originally intended for scientific visualization led to difficulties, particularly the need to identify item-level data during picking. Otherwise, the modular design provided many of the facilities needed for this project, for example the network visualization pipeline, interaction and event handling mechanisms, and integration with Qt. The new scene-graph API being developed for VTK may prove to be a superior choice for future information visualization applications, none the less the final result was the development of a fully operational network visualization tool.

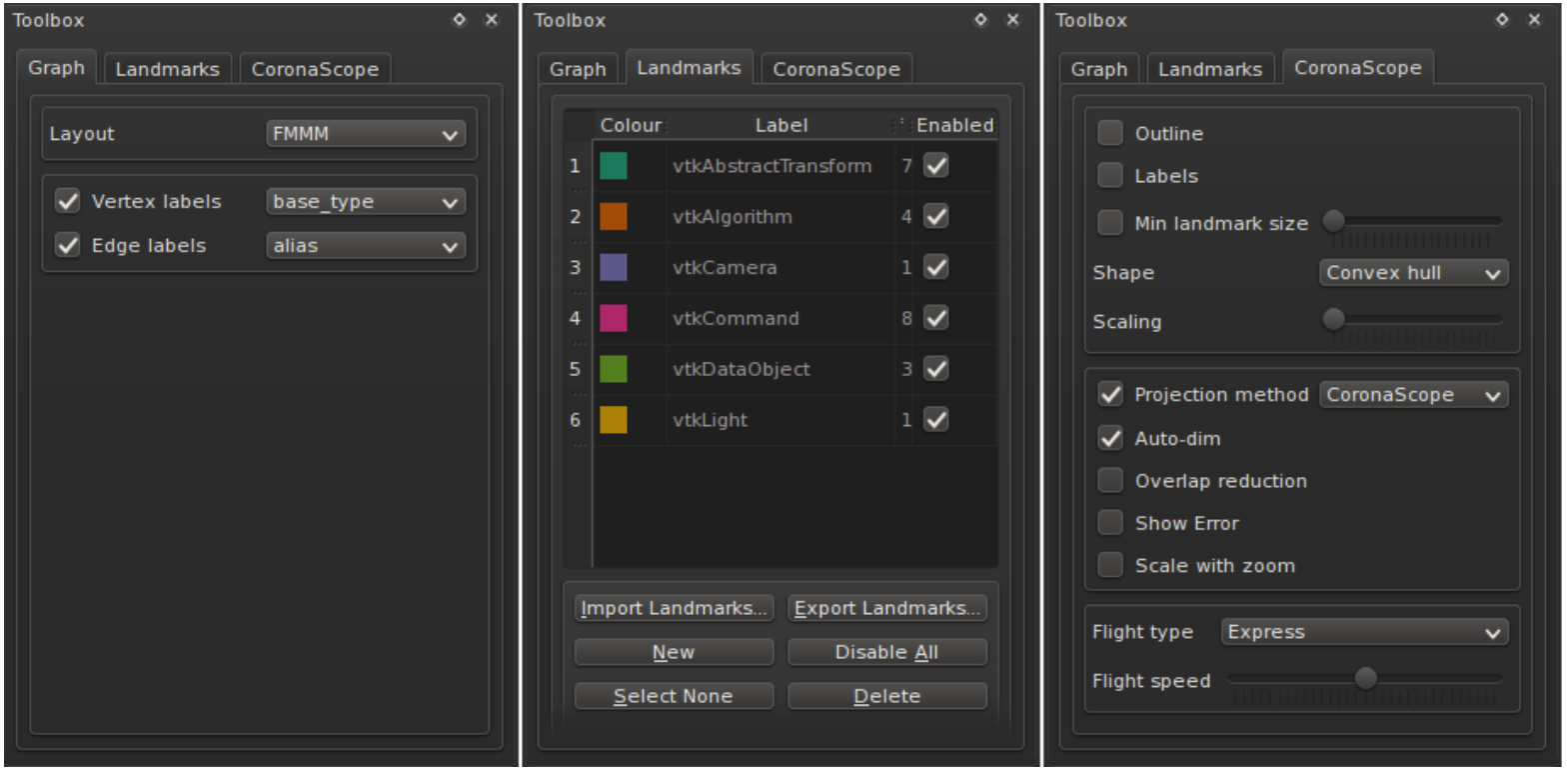

Figure 6.7: Screen shots of the CoronaScope toolbox. The *Graph* panel contains options for layout and node/edge labelling; the *Landmarks* panel is used to import, export, and edit sets of landmarks and acts as a colour-class legend; and, the *CoronaScope* panel provides configuration options for the landmarks overlay, off-screen visualization, and animation parameters.

Chapter 7

Conclusion

In this final chapter I summarize the problem of network navigation, reflect on the strengths and limitations of the cognitive theory-based approach, and the resulting design of an overlay of landmarks. In conclusion, I offer some recommendations for future directions arising from this work.

7.1 Network Navigation

Information visualization supports *exploratory analysis* of relational data by allowing the user to interactively reconfigure and refine the view. A *network* is a visual representation of a graph that is particularly effective for understanding paths formed by sequences of nodes and edges, and *navigation* is the act of selecting and following such paths. There are cognitive costs incurred during navigation, partly due to the potential confusion arising from false nodes that appear where two distinct edges cross. Also, the limitations of display dimensions and resolution with respect to increasingly large networks mean that visual clutter is inevitable, with the potential for users to become lost in "desert fog". To counter this problem, pan and zoom allows the user to adjust the view, zooming in to read the fine details of nodes and edges, and zooming out to understand these local features within the global context, though doing so entails the additional cognitive costs of maintaining object-association during movements. These costs add up to form a "gulf of evaluation" in comprehending the network, and a "gulf of formation" whereby the information needed to form conceptual goals, for example "where do I go next?", is not available, leading to task breakdown.

7.2 Theory and Approach

Inspired by MacEachren's multi-level, multi-discipline approach to a theory of map comprehension I set out theories and evidence related to the comprehension of networks from various points of view:

perceptual concerns included models of visual attention, and empirical studies of low-level path-following tasks;

cognitive models of graph and map comprehension, and theories of internal representations (schemata); and,

semantic attributes of networks induced by layout algorithms based on purely topological data.

Synthesizing these concepts with models of information visualization I highlighted the need for *congruence*: a logical organization of the display that reflects the expected structure of the user's internal representation. This becomes particularly relevant in the case of networks where graph layout algorithms based on topology optimize the aesthetic appearance of the network, often failing to account for its semantic content. Like Agrawala and Stolte's route-map system [2], visualization designs should be based on an understanding of the systematic distortions in the user's mental map of an environment. This led me to establish a link between the internal processes used in real-world navigation, and those we might expect to be employed during navigation of a network.

7.3 An Overlay of Landmarks

Landmarks were proposed as a means of providing a framework of navigation references, just as they do in both urban, and virtual environments. I defined a landmark in a network as a meaningful subgraph representing a functional unit in the semantic domain of the data, and proposed a set of guidelines for selecting and depicting landmarks. The aim was to provide a set of visual references that are available throughout scale-space, by setting a minimum size regardless of scale, and through off-screen visualization. Because the landmarks provide semantic information rather than merely acting as structural markers, they promote congruence by acting as a high-level semantic overview of the network. The framework of landmarks enabled the provision of some support for the

specific key tasks of *following long paths in scale-space* using an edge-based animated pan and zoom, and *revisiting*, an essential component of exploration tasks.

The proposed new designs for overlaid navigational tools support were implemented in a desktop application, built on a pipeline-based framework. The framework was originally designed specifically for scientific visualization but later adapted with new features to support information visualization data types. The assumption implicit in the pipeline model that the final view is an interpolated space led to difficulties in identifying item-level data during picking. However, a fully functional prototype application was successfully produced by working round the problem, made possible by the relatively small number of landmarks. Two of the new components now form part of the public release of the Visualization Tool Kit, and acceptance by a user community is one way in which a design can be validated [88].

7.4 Looking Back

During analysis of the new designs some strengths and limitations were identified. Due to the semantic dimension of the explicit landmarks they immediately provided a general overview of the network, indicating some recognizable features, potentially extending beyond individual nodes to functional subgraphs or motifs. With careful use of shape, colour, and transparency it was possible to create additional *visual levels*, without obfuscating the primary network representation. The creation of visual levels relied here on transparency, and could benefit from a perception-based transparency measure since different hues produced vary levels of intensity for the same level of opacity. The concept of visual levels rarely features in the information visualization literature, though efficient deployment of visual attention is key to achieving efficiency. In cartography visual levels are exploited to produce information-dense displays, and this could become particularly relevant as display densities increase.

The design of an off-screen visualization to provide global awareness of landmarks throughout scale-space was more complicated, due to the limitations in our understanding of how users reason about off-screen space. The implementation described in this thesis was predicated on models of real-world navigation, and while theories of how schemata visualization evolve from a more general navigation schema are evidence of a connection between the two domains, there are also some significant differences. Not least is that viewing a network is not embodied in the same way that moving in the

real world, or even a 3-D virtual environment, being more like reading a map: a scaled representation of some ‘other’ place.

Although some specific navigation task support was demonstrated (path-following and revisiting), many parts of the design would be challenging to validate empirically, due to the expectation that users have some network visualization expertise, knowledge about the data domain, and because many of the expected benefits of enhanced navigational knowledge may require the development of longer-term memory structures that take time to develop, potentially over multiple analysis sessions.

7.5 Looking Forward

Being predicated on cognitive theories the work here indicated a lack of understanding of the user’s internal representation, and the exact information required to support navigation tasks in any one moment. Models of salience are insufficiently tailored to network representation, and perceptual evidence from network user studies is limited. These limitations suggest a need to investigate network comprehension but to go beyond short-term, localized path-following tasks, and instances of domain-specific applications. The multi-level approach (low-level vision, cognitive theories, semiotics) to map comprehension espoused by MacEachren [78] combines theories of low-level vision, comprehension, and semiotics: a similar approach to a unified model of network navigation would be a positive step. Particularly useful would be a functional model of salience, to predict how visual attention is deployed during network navigation, linked with a model of network features (nodes, edges, subgraphs) as they appear in scale-space. One possibility is that groups of landmarks may form a hierarchy, as noted in Section 4.2.2.2. The initial selection of landmarks is crucial, and I presented some guidelines, but the methods suggested often require a great deal of work ‘up-front’ to identify the appropriate technique. To overcome this problem CoronaScope would benefit from the automated selection of landmarks, based on the models just described.

A dimension of landmarks not explored in this thesis was lifetime, as the aim was to guarantee a stable set of visual references. However, during animated or manual pan and zoom operations for example, it may be helpful to provide temporary landmarks to assist with sub-tasks. While manual annotation is a step in this direction, as implemented it was not easy to create and manage temporary landmarks as this represented a break in the primary activity. A more analysis-focussed approach (such as Shrinivasan and

van Wijk's "knowledge view" [114]), indicating search results in-situ by for example, matching motifs, or attribute-based filtering, could extend the ideas presented in this thesis to a wider range of tasks.

In summary, separation of navigation concerns from those of representation has been beneficial in recognizing the limitations of topology-based layout algorithms, leading to consideration of the navigation task with respect to both the appearance of networks in scale-space, and the user's internal model of that space. An ongoing challenge is the development of a deeper understanding of both external and internal representations, to reduce the gulfs of evaluation and formation, and improve congruence.

Bibliography

[1] J. Abello, F. van Ham, and N. Krishnan. ASK-GraphView: a large scale graph visualization system. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):669 – 676, 2006.

[2] M. Agrawala and C. Stolte. Rendering effective route maps: Improving usability through generalization. In *Proceedings of the annual conference on Computer Graphics*, pages 241 – 249. ACM Press, 2001.

[3] A. Ahmed and P. Eades. Automatic camera path generation for graph navigation in 3D. In *Proceedings of the Asia-Pacific symposium on Information Visualization*, pages 27 – 32. Australian Computer Society, 2005.

[4] R. Alberich, J. Miro-Julia, and F. Rossello. Marvel universe looks almost like a real social network. *eprint arXiv:cond-mat/0202174*, 2001.

[5] B. Alper, N. H. Riche, G. Ramos, and M. Czerwinski. Design study of LineSets, a novel set visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, pages 2259 – 2267, 2011.

[6] D. Auber. Tulip: a huge graph visualisation framework. In P. Mutzel and M. Jünger, editors, *Graph Drawing Softwares*, Mathematics and Visualization, pages 105 – 126. Springer-Verlag, 2003.

[7] M. Balzer and O. Deussen. Level-of-detail visualization of clustered graph layouts. In *Proceedings of the international Asia-Pacific symposium on Visualization*, pages 133 – 140. IEEE Computer Society Press, 2007.

[8] M. Balzer, A. Noack, O. Deussen, and C. Lewerentz. Software landscapes: Visualizing the structure of large software systems. In *Proceedings of the Symposium on Visualization*, pages 261 – 266. Eurographics Association, 2004.

[9] G. Bartel, C. Gutwenger, K. Klein, and P. Mutzel. An experimental evaluation of multilevel layout methods. In U. Brandes and S. Cornelsen, editors, *Graph Drawing*, volume 6502 of *Lecture Notes in Computer Science*, pages 80 – 91. Springer, 2011.

[10] L. Bartram, B. Cheung, and M. Stone. The effect of colour and transparency on the perception of overlaid grids. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1942 – 1948, 2011.

[11] M. Bastian, S. Heymann, and M. Jacomy. Gephi: an open source software for exploring and manipulating networks. In *International AAAI conference on Weblogs and Social Media*. AAAI, 2009.

[12] P. Baudisch and R. Rosenholtz. Halo: a technique for visualizing off-screen locations. In *Proceedings of the annual conference on Human Factors in Computing Systems*, pages 481 – 488. ACM Press, 2003.

[13] P. Baudisch, D. Tan, M. Collomb, D. Robbins, K. Hinckley, M. Agrawala, S. Zhao, and G. Ramos. Phosphor: explaining transitions in the user interface using afterglow effects. In *Proceedings of the annual symposium on User Interface Software and Technology*, pages 169–178. ACM Press, 2006.

[14] J. Bertin. *Semiology of Graphics: Diagrams Networks Maps*. Esri Press, 2010.

[15] A. Bezerianos, P. Dragicevic, J.-D. Fekete, J. Bae, and B. Watson. GeneaQuilts: a system for exploring large genealogies. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1073 – 1081, 2010.

[16] M. Bostock, V. Ogievetsky, and J. Heer. D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[17] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163 – 177, 2001.

[18] C. A. Brewer. ColorBrewer2. `http://www.ColorBrewer.org`. Accessed 10th February, 2013.

[19] D. M. Butler, J. C. Almond, R. D. Bergeron, K. W. Brodlie, and R. B. Haber. Visualization reference models. In *Proceedings of IEEE Visualization*, pages 337 – 342. IEEE Computer Society, 1993.

[20] H. Byelas and A. Telea. Visualization of areas of interest in software architecture diagrams. In *Proceedings of the ACM symposium on Software visualization*, pages 105 – 114. ACM Press, 2006.

[21] S. K. Card, J. D. MacKinlay, and B. Shneiderman, editors. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, 1999.

[22] S. M. Casner. Task-analytic approach to the automated design of graphic presentations. *ACM Transactions on Graphics*, 10(2):111 – 151, 1991.

[23] E. Chi. A taxonomy of visualization techniques using the data state reference model. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 69 – 75. IEEE Computer Society, 2000.

[24] E. Chi and J. Riedl. An operator interaction framework for visualization systems. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 63 – 70. IEEE Computer Society, 1998.

[25] M. Chimani, C. Gutwenger, M. Jünger, G. W. Klau, and P. M. K. Klein. The open graph drawing framework (OGDF). In R. Tamassia, editor, *Handbook of Graph Drawing and Visualization*. CRC Press, To appear.

[26] A. Cockburn, A. Karlson, and B. B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys*, 41(1):2:1 – 2:31, 2008.

[27] C. Collins, G. Penn, and S. Carpendale. Bubble Sets: Revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1009 – 1016, 2009.

[28] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, second edition, 2001.

[29] D. A. Cox, J. S. Chugh, C. Gutwin, and S. Greenberg. The usability of transparent overview layers. In *Conference summary on Human Factors in Computing Systems*, pages 301 – 302. ACM Press, 1998.

[30] B. Craft and P. A. Cairns. Beyond guidelines: What can we learn from the visual information seeking mantra? In *Proceedings of the 9th International Conference on Information Visualization*, pages 110 – 118. IEEE Computer Society, 2005.

[31] R. P. Darken and J. L. Sibert. A toolset for navigation in virtual environments. In *Proceedings of the 6th annual ACM symposium on User Interface Software and Technology*, pages 157 – 165. ACM Press, 1993.

[32] E. Dengler and W. B. Cowan. Human perception of laid-out graphs. In *Proceedings of the 6th international symposium on Graph Drawing*, pages 441 – 443. Springer-Verlag, 1998.

[33] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry: Theory and Applications*, 4:235 – 282, 1994.

[34] Digia Oyj. Qt. `http://qt.digia.com`. Accessed 10th February, 2013.

[35] D. P. Dobkin, E. R. Gansner, E. Koutsofios, and S. C. North. Implementing a general-purpose edge router. In *Proceedings of the 5th international symposium on Graph Drawing*, pages 262 – 271. Springer-Verlag, 1997.

[36] D. J. Duke. Modular techniques in information visualization. In *Proceedings of the Australian symposium in Information Visualization*, volume 9, pages 11 – 18. Australian Computer Society, 2001.

[37] T. Dwyer. Scalable, versatile and simple constrained graph layout. *Computer Graphics Forum*, 28(3):991 – 998, 2009.

[38] T. Dwyer, Y. Koren, and K. Marriott. IPSep-CoLa: an incremental procedure for separation constraint layout of graphs. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):821 – 828, 2006.

[39] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149 – 160, 1984.

[40] P. Eades and Q.-W. Feng. Multilevel visualization of clustered graphs. In *Proceedings of the 4th international symposium on Graph Drawing*, pages 101 – 112. Springer-Verlag, 1996.

[41] Eclipse Foundation. Eclipse IDE. `http://www.eclipse.org`. Accessed 10th February, 2013.

[42] S. G. Eick. Engineering perceptually effective visualizations for abstract data. In *Scientific Visualization*, pages 191 – 210. IEEE Computer Society Press, 1994.

[43] W. D. Ellis. *A Source Book of Gestalt Psychology*. Humanities Press, 1950.

[44] K. M. Fairchild, S. E. Poltrock, and G. W. Furnas. SemNet: Three-dimensional graphic representations of large knowledge bases. In S. K. Card, J. D. MacKinlay, and B. Shneiderman, editors, *Readings in Information Visualization*, pages 190 – 206. Morgan Kaufmann, 1999.

[45] J.-D. Fekete. The InfoVis toolkit. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 167 – 174. IEEE Computer Society, 2004.

[46] J.-D. Fekete, G. Grinstein, and C. Plaisant. The history of infovis benchmark dataset. `http://www.cs.umd.edu/hcil/iv04contest/register.html`. Accessed 10th February, 2013.

[47] M. Frisch and R. Dachselt. Off-screen visualization techniques for class diagrams. In *Proceedings of the 5th international symposium on Software Visualization*, pages 163 – 172. ACM Press, 2010.

[48] T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Software Practice and Experience*, 21(11):1129 – 1164, 1991.

[49] G. W. Furnas. Effective view navigation. In *Proceedings of the annual conference on Human Factors in Computing*, pages 367 – 374. ACM Press, 1997.

[50] G. W. Furnas and B. B. Bederson. Space-scale diagrams: Understanding multiscale interfaces. In *Proceedings of the annual conference on Human Factors in Computing Systems*, pages 234 – 241. ACM Press, 1995.

[51] E. Gansner, Y. Hu, and S. Kobourov. GMap: Visualizing graphs and clusters as maps. In *IEEE Pacific Visualization Symposium*, pages 201 – 208. IEEE Computer Society, 2010.

[52] E. R. Gansner and S. C. North. An open graph visualization system and its applications to software engineering. *Software: Practice and Experience*, 30(11):1203 – 1233, 2000.

[53] S. Ghani, N. H. Riche, and N. Elmqvist. Dynamic insets for context-aware graph navigation. *Computer Graphics Forum*, 30(3):861 – 870, 2011.

[54] M. Ghoniem, J.-D. Fekete, and P. Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. In *Proceedings of the*

IEEE Symposium on Information Visualization, pages 17 – 24. IEEE Computer Society Press, 2004.

[55] S. Gustafson, P. Baudisch, C. Gutwin, and P. Irani. Wedge: Clutter-free visualization of off-screen locations. In *Proceedings of the annual conference on Human Factors in Computing Systems*, pages 787 – 796. ACM Press, 2008.

[56] S. G. Gustafson and P. P. Irani. Comparing visualizations for tracking off-screen moving targets. In *Proceedings of the annual conference on Human Factors in Computing Systems*, pages 2399 – 2404. ACM Press, 2007.

[57] R. B. Haber and D. McNabb. Visualization idioms: a conceptual model for scientific visualization systems. In G. Nielson and B. Shriver, editors, *Visualization in Scientific Computing*. IEEE Computer Society Press, 1990.

[58] S. Hachul and M. Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In *Proceedings of the 12th international conference on Graph Drawing*, pages 285 – 295. Springer-Verlag, 2004.

[59] S. Haroz and D. Whitney. How capacity limits of attention influence information visualization effectiveness. *IEEE Transactions on Visualization and Computer Graphics (Early Access)*, 18(12):2402 – 2410, 2012.

[60] C. G. Healey and J. T. Enns. Attention and visual memory in visualization and computer graphics. *IEEE Transactions on Visualization and Computer Graphics (Early Access)*, 18(7):1180 – 1188, 2012.

[61] J. Heer, S. K. Card, and J. A. Landay. Prefuse: a toolkit for interactive information visualization. In *Proceedings of the annual conference on Human Factors in Computing Systems*, pages 421 – 430. ACM Press, 2005.

[62] N. Henry and J.-D. Fekete. MatrixExplorer: a dual-representation system to explore social networks. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):677–684, 2006.

[63] I. Herman, M. Delest, and G. Melançon. Tree visualisation and navigation clues for information visualisation. *Computer Graphics Forum*, 17(2):153 – 165, 1998.

[64] I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: a survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24 – 43, 2000.

[65] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12:741 – 748, 2006.

[66] W. Huang and P. Eades. How people read graphs. In *Proceedings of the Asia-Pacific Symposium on Information Visualization*, pages 51 – 58. Australian Computer Society, 2005.

[67] W. Huang, P. Eades, and S.-H. Hong. A graph reading behaviour: geodesic-path tendency. In *IEEE Pacific Visualization Symposium*, pages 137 – 144. IEEE Computer Society, 2009.

[68] W. Huang, S.-H. Hong, and P. Eades. Effects of sociogram drawing conventions and edge crossings in social network visualization. *Journal of Graph Algorithms Applied*, 11(2):397 – 429, 2007.

[69] P. Irani, C. Gutwin, and X. D. Yang. Improving selection of off-screen targets with hopping. In *Proceedings of the annual conference on Human Factors in Computing Systems*, pages 299 – 308. ACM Press, 2006.

[70] S. Jul and G. W. Furnas. Critical zones in desert fog: Aids to multiscale navigation. In *Proceedings of the 11th annual ACM symposium on User Interface Software and Technology*, pages 97 – 106. ACM Press, 1998.

[71] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7 – 15, 1989.

[72] M. Kanehisa and S. Goto. KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, 28(1):27 – 30, 2000.

[73] H. Lam. A framework of interaction costs in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1149 – 1156, 2008.

[74] B. Lee, C. Plaisant, C. S. Parr, J.-D. Fekete, and N. Henry. Task taxonomy for graph visualization. In *Proceedings of the AVI workshop on BEyond Time and Errors*, pages 1 – 5. ACM Press, 2006.

[75] Z. Liu, N. Nersessian, and J. Stasko. Distributed cognition as a theoretical framework for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1173 – 1180, 2008.

[76] Z. Liu and J. Stasko. Mental models, visual reasoning and interaction in information visualization: a top-down perspective. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):999 – 1008, 2010.

[77] K. Lynch. *The Image of the City*. MIT Press, 1960.

[78] A. M. MacEachren. *How Maps Work: Representation, Visualization, and Design*. Guilford Press, 1995.

[79] J. D. MacKinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110 – 141, 1986.

[80] A. Maedche and S. Staab. Mining ontologies from text. In *Proceedings of the 12th European workshop on Knowledge Acquisition, Modeling and Management*, pages 189 – 202. Springer-Verlag, 2000.

[81] K. Marriott, H. Purchase, M. Wybrow, and C. Goncu. Memorability of visual features in network diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2477 – 2485, 2012.

[82] J. May, P. J. Barnard, and A. Blandford. Using structural descriptions of interfaces to automate the modelling of user cognition. *User Modeling and User-Adapted Interaction*, 3(1):27 – 64, 1993.

[83] B. H. McCormick. Visualization in scientific computing – a synopsis. *Computer Graphics and Applications*, 7(7):6 – 70, 1987.

[84] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824 – 827, 2002.

[85] T. Moscovich, F. Chevalier, N. Henry, E. Pietriga, and J.-D. Fekete. Topology-aware navigation in large networks. In *Proceedings of the annual conference on Human Factors in Computing Systems*, pages 2319 – 2328. ACM Press, 2009.

[86] S. Mukherjea and Y. Hara. Focus+context views of world-wide web nodes. In *Proceedings of the 8th ACM conference on Hypertext*, pages 187 – 196. ACM Press, 1997.

[87] T. Munzner. *Interactive Visualization of Large Graphs and Networks*. PhD thesis, Stanford University, 2000.

[88] T. Munzner. A nested model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921 – 928, 2009.

[89] T. Munzner, F. Guimbretière, and G. Robertson. Constellation: a visualization tool for linguistic queries from MindNet. In *Proceedings of the 1999 IEEE Symposium on Information Visualization*, pages 132 – 135. IEEE Computer Society Press, 1999.

[90] T. Munzner, F. Guimbretière, S. Tasiran, L. Zhang, and Y. Zhou. TreeJuxtaposer: Scalable tree comparison using focus+context with guaranteed visibility. *ACM Transactions on Graphics*, 22(3):453 – 462, 2003.

[91] C. Myers and D. Duke. A map of the heap: Revealing design abstractions in runtime structures. In *Proceedings of the 5th international symposium on Software Visualization*, pages 63 – 72. ACM Press, 2010.

[92] D. A. Norman. *The Design of Everyday Things*. Doubleday, 1988.

[93] G. Partridge, M. Nezhadasl, P. Irani, and C. Gutwin. A comparison of navigation techniques across different types of off-screen navigation tasks. In *Proceedings of the 11th IFIP International conference on Human-Computer Interaction*, pages 716 – 721. Springer, 2007.

[94] S. Pinker. A theory of graph comprehension. In R. Freedle, editor, *Artificial intelligence and the future of testing*, pages 73 – 126. Psychology Press, 1990.

[95] C. Plaisant, J. Grosjean, and B. B. Bederson. SpaceTree: Supporting exploration in large node link trees, design evolution and empirical evaluation. In *Proceedings of the IEEE symposium on Information Visualization*, pages 57 – 65. IEEE Computer Society Press, 2002.

[96] A. J. Pretorius and J. J. van Wijk. Bridging the semantic gap: Visualization of transition graphs with user-defined diagrams. *Computer Graphics and Applications*, 27(5):58 – 66, 2007.

[97] A. J. Pretorius and J. J. van Wijk. Visual inspection of multivariate graphs. *Computer Graphics Forum*, 27(3):967 – 974, 2008.

[98] H. C. Purchase. Which aesthetic has the greatest effect on human understanding? In *Proceedings of the 5th international symposium on Graph Drawing*, pages 248 – 261. Springer-Verlag, 1997.

[99] H. C. Purchase, E. Hoggan, and C. Görg. How important is the "mental map": An empirical investigation of a dynamic graph layout algorithm. In *Proceedings of the 14th international conference on Graph Drawing*, pages 184 – 195. Springer-Verlag, 2007.

[100] P. Quinn, A. Cockburn, J. Indratmo, and C. Gutwin. An investigation of dynamic landmarking funcitons. In *Proceedings of the international conference on Advanced Visual Interfaces*, pages 322 – 325. ACM Press, 2008.

[101] R. A. Rensink. The management of visual attention in graphic displays. In C. Roda, editor, *Human Attention in Digital Environments*, chapter 3. Cambridge University Press, 2011.

[102] A. E. Richardson, D. R. Montello, and M. Hegarty. Spatial knowledge acquisition from maps and from navigation in real and virtual environements. *Memory and Cognition*, 27(4):741 – 750, 1999.

[103] N. H. Riche and T. Dwyer. Untangling Euler diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1090 – 1099, 2010.

[104] G. Robertson, S. K. Card, and J. D. MacKinlay. The cognitive coprocessor architecture for interactive user interfaces. In *Proceedings of the 2nd annual ACM SIGGRAPH symposium on User Interface Software and Technology*, pages 10 – 18. ACM Press, 1989.

[105] P. Saffrey and H. C. Purchase. The "mental map" versus "static aesthetic" compromise in dynamic graphs: a user study. In *Proceedings of the 9th conference on Australasian user interface*, pages 85 – 93. Australian Computer Society, Inc., 2008.

[106] R. Sanderson, P. Ciccarese, and H. van de Sompel. W3C open annotation data model. `http://www.openannotation.org/spec/core/20130208/index.html`. Accessed 10th February, 2013.

[107] M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. In *Proccedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 83 – 91. ACM, 1992.

[108] F. Schreiber, T. Dwyer, K. Marriott, and M. Wybrow. A generic algorithm for layout of biological networks. *BMC Bioinformatics*, 10:375, 2009.

[109] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit*. Kitware Inc., fourth edition, 2006.

[110] D. Selassie, B. Heller, and J. Heer. Divided edge bundling for directional network data. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2354 – 2363, 2011.

[111] B. Shneiderman. Tree visualization with tree-maps: 2-D space-filling approach. *ACM Transactions on Computer Graphics*, 11(1), 1992.

[112] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, page 336. IEEE Computer Society, 1996.

[113] B. Shneiderman and A. Aris. Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):733 – 740, 2006.

[114] Y. B. Shrinivasan and J. J. van Wijk. Supporting the analytical reasoning process in information visualization. In *Proceedings of the annual conference on Human Factors in Computing Systems*, pages 1237 – 1246. ACM Press, 2008.

[115] J. G. Siek, L.-Q. Lee, and A. Lumsdaine. *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley, 2001.

[116] M. E. Sorrows and S. C. Hirtle. The nature of landmarks for real and electronic spaces. In *Proceedings of the international conference on Spatial Information Theory: Cognitive and Computational Foundations of Geographic Information Science*, pages 37 – 50. Springer-Verlag, 1999.

[117] R. Spence. *Information Visualization: Design for Interaction*. Prentice Hall, second edition, 2007.

[118] T. C. Sprenger, R. Brunella, and M. H. Gross. H-BLOB: a hierarchical visual clustering method using implicit surfaces. In *Proceedings of the IEEE conference on Visualization*, pages 61 – 68. IEEE Computer Society Press, 2000.

[119] A. H. Squillacote. *The ParaView Guide: A Parallel Visualization Application*. Kitware Inc., second edition, 2007.

[120] F. Steinbrückner and C. Lewerentz. Representing development history in software cities. In *Proceedings of the 5th international symposium on Software Visualization*, pages 193 – 202. ACM Press, 2010.

[121] M. Stone and L. Bartram. Alpha, contrast and the perception of visual metadata. In *16th Color Imaging Conference: Color Science and Engineering Systems, Technologies, and Applications*, pages 355 – 359. The Society for Imaging Science and Technology, 2008.

[122] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics*, 11(2):109 – 125, 1989.

[123] C. Tominski, J. Abello, and H. Schumann. CGV – an interactive graph visualization system. *Computers and Graphics*, 33(6):660 – 678, 2009.

[124] M. Tory and T. Möller. Rethinking visualization: a high-level taxonomy. In *Proceedings of the IEEE symposium on Information Visualization*, pages 151 – 158. IEEE Computer Society, 2004.

[125] E. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.

[126] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[127] B. Tversky. Cognitive maps, cognitive collages, and spatial mental models. In *Spatial Information Theory A Theoretical Basis for GIS*, pages 14 – 24. Springer-Verlag, 1993.

[128] B. Tversky, J. B. Morrison, and M. Betrancourt. Animation: Can it facilitate? *International Journal of Human-Computer Studies*, 57:247 – 262, 2002.

[129] B. Tversky, J. Zacks, P. U. Lee, and J. Heiser. Lines, blobs, crosses and arrows: Diagrammatic communication with schematic figures. In *Proceedings of the 1st international conference on the Theory and Application of Diagrams*, pages 221 – 230. Springer-Verlag, 2000.

[130] F. van Ham and A. Perer. "Search, show context, expand on demand": Supporting graph exploration with degree-of-interest. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):953 – 960, 2009.

[131] J. J. van Wijk and W. A. A. Nuij. A model for smooth viewing and navigation of large 2D information spaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(4):447 – 458, 2004.

[132] N. G. Vinson. Design guidelines for landmarks to support navigation in virtual environments. In *Proceedings of the annual conference on Human Factors in Computing Systems*, pages 278 – 285. ACM Press, 1999.

[133] T. von Landesberger, M. Görner, R. Rehner, and T. Schreck. A system for interactive visual analysis of large graphs using motifs in graph editing and aggregation. In *Proceedings of the Vision, Modeling and Visualization Workshop*, pages 331 – 340. Eurographics Association, 2009.

[134] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, and D.W.Fellner. Visual analysis of large graphs: State-of-the-art and future research challenges. *Computer Graphics Forum*, 30(6):1719 – 1749, 2011.

[135] M. O. Ward and J. Yang. Interaction spaces in data and information visualization. In *VisSym*, pages 137 – 145. Eurographics Association, 2004.

[136] C. Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, second edition, 2004.

[137] C. Ware. *Visual Thinking: for Design*. Morgan Kaufmann, 2008.

[138] C. Ware, H. Purchase, L. Colpoys, and M. McGill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103 – 110, 2002.

[139] B. Wylie and J. Baumes. A unified toolkit for information and scientific visualization. In *Proceedings of Visualization and Data Analysis*, volume 7243. SPIE, 2009.

[140] J. S. Yi, Y. Kang, J. Stasko, and J. Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6), 2007.

[141] P. T. Zellweger, J. D. MacKinlay, L. Good, M. Stefik, and P. Baudisch. City Lights: Contextual views in minimal space. In *Extended abstracts on Human Factors in Computing Systems*, pages 838 – 839. ACM Press, 2003.

[142] M. X. Zhou. Automated visual discourse synthesis: Coherence, versatility, and interactivity. In *Conference summary on Human Factors in Computing Systems*, pages 76 – 77. ACM Press, 1998.