Object-centric Temporal Navigation for Dynamic Information Visualizations

Brittany Kondo

Faculty of Science

University of Ontario Institute of Technology

A thesis submitted in conformity with the requirements for

Master of Science

2014

Acknowledgements

I am grateful to Dr. Christopher Collins for his guidance and advice during my masters work. I would like to thank the members of the vialab who provided their advice and support. Lastly, I would like to offer special thanks to Hrim Mehta for her contributions to this work.

List of Publications

The following publications resulted from work presented in this thesis:

Brittany Kondo, Hrim Mehta, Christopher Collins. Glidgets: Interactive Glyphs for Exploring Dynamic Graphs. Graphics, Animation and New Media Poster Session (GRAND'14). 2014.

Brittany Kondo, Christopher Collins. DimpVis: Exploring Time-varying Information Visualizations by Direct Manipulation. (To appear) IEEE Transactions on Visualization and Computer Graphics (VIS'14). 2014.

Brittany Kondo, Hrim Mehta, Christopher Collins. Glidgets: Interactive Glyphs for Exploring Dynamic Graphs. (To appear) In Proc. of the IEEE Information Visualization Conference (Poster Session VIS'14). 2014.

Supplemental Material

The following supplemental materials can be accessed online:

DimpVis
http://vialab.science.uoit.ca/portfolio/dimpvis/

Glidgets
http://vialab.science.uoit.ca/portfolio/glidgets/

Abstract

We introduce object-centric temporal navigation for exploring time-varying information visualizations. In our approach, navigation through time is controlled by interacting directly with any data item, enabling simultaneous exploration of the time dimension while focusing on the changing item of interest. Subtle visualizations of a data item's temporal trend are provided to guide navigation. To demonstrate how object-centric navigation can be designed for different types of dynamic visualizations, we created two techniques: DimpVis, for exploring changing visual variables in different types of varying information visualizations, and Glidgets, an interactive glyph-based technique for exploring topological changes in dynamic graphs. Both techniques enable intuitive investigation of spatial queries. For example, using DimpVis to answer "Was this bar ever at 500?" in a time-varying bar chart, one simply has to drag the bar to that height. Comparative task-based evaluations revealed that DimpVis for the scatter plot was quantitatively competitive with the time slider and small multiples. Additionally, both Glidgets and DimpVis were overall subjectively preferred over the time slider by participants.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Interactive information visualizations exploit dynamic visual representations of data, using interaction to expedite exploration and navigation [58]. Interactivity can alleviate restrictive aspects of static representations, enhancing human cognition [58]. When datasets are large or dense, interactivity presents a solution for relieving cognitive load, provoking user engagement, immersion and enjoyment [24]. The relationship between interaction and inquiry, identified by Pike et al. as "interaction is the inquiry," implies the dominant role of interaction for knowledge construction [41].

Dynamic data changes across some evolving dimension, such as time. Visualizing time-varying data involves representing the data, at each moment in time. Analysts are interested in detecting and characterizing changes in varying data, and interaction techniques can help support analysis and exploration of the time dimension. Temporally evolving changes can be observed at a low-level, such as examining how an individual data item changes, or at a high-level, such as observing how the entire dataset changes. Low-level changes directly contribute to higher level changes. For example, adding edges can lead to cluster formation, in dynamic graphs. In this case, it's important to understand how graph elements are changing at a low level, in order to identify and describe high level changes [6].

Many types of real world data, such as census statistics, sales revenues, stock market prices, and twitter feeds, change over time. Familiar chart types, such as bar charts and scatter plots can be used to represent this time-varying data. For instance, the popular Gapminder system [46] uses animated bubble charts to visualize and explore time-varying population statistics. Additionally, time-varying datasets containing connectivity relationships among data elements, such as social networks are typically visualized using graph visualizations. Dynamic graphs are readily used in important domains such as social

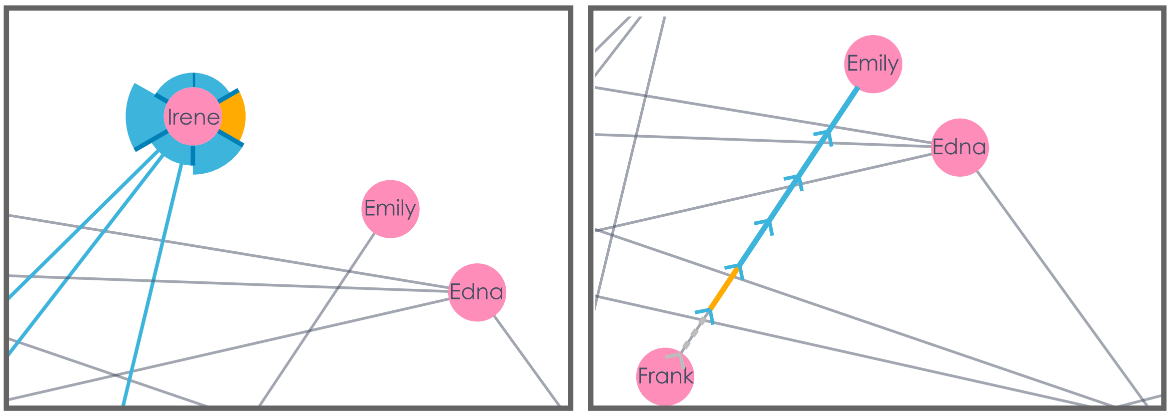

Fig. 1.1 The interactive node (left) and edge (right) glyphs showing when the element is present (blue) and absent (grey) over time. Each segment of the glyph represents a single time point. In the node glyph, the varying height of segments represents relative node degree.

network analysis, biology (e.g., brain connectivity analysis [18]), and computer network security.

We introduce *object-centric temporal navigation*, an interaction technique for exploring different types of time-varying information visualizations. Navigation through time is controlled by interacting directly with any data item of interest; thus the data item becomes the focus and control for temporal navigation. Object-centric navigation is intended for low-level exploration of an individual data item's changing values.

In this work, we present two object-centric temporal navigation techniques: *DimpVis*, for exploring changing visual variables in common types of varying information visualizations, and *Glidgets*, an interactive glyph-based interface for exploring topological changes in dynamic graphs. Both techniques enable intuitive investigation of spatial queries. For example, using DimpVis to answer "Was this bar ever at 500?" in a time-varying bar chart, one simply has to drag the bar to that height. If a moment in time exists when the bar is at the height, the visualization is moved to that time. The interaction is guided by visual paths which reveal how a selected data item changes through the time dimension. Alternatively, using Glidgets to answer "Are two nodes ever connected over time?", one can first draw a line connecting the nodes and examine an edge glyph that reveals if and when the nodes were connected (Figure 1.1(right)). Furthermore, navigation directly to any time point can be done by dragging along the glyph.

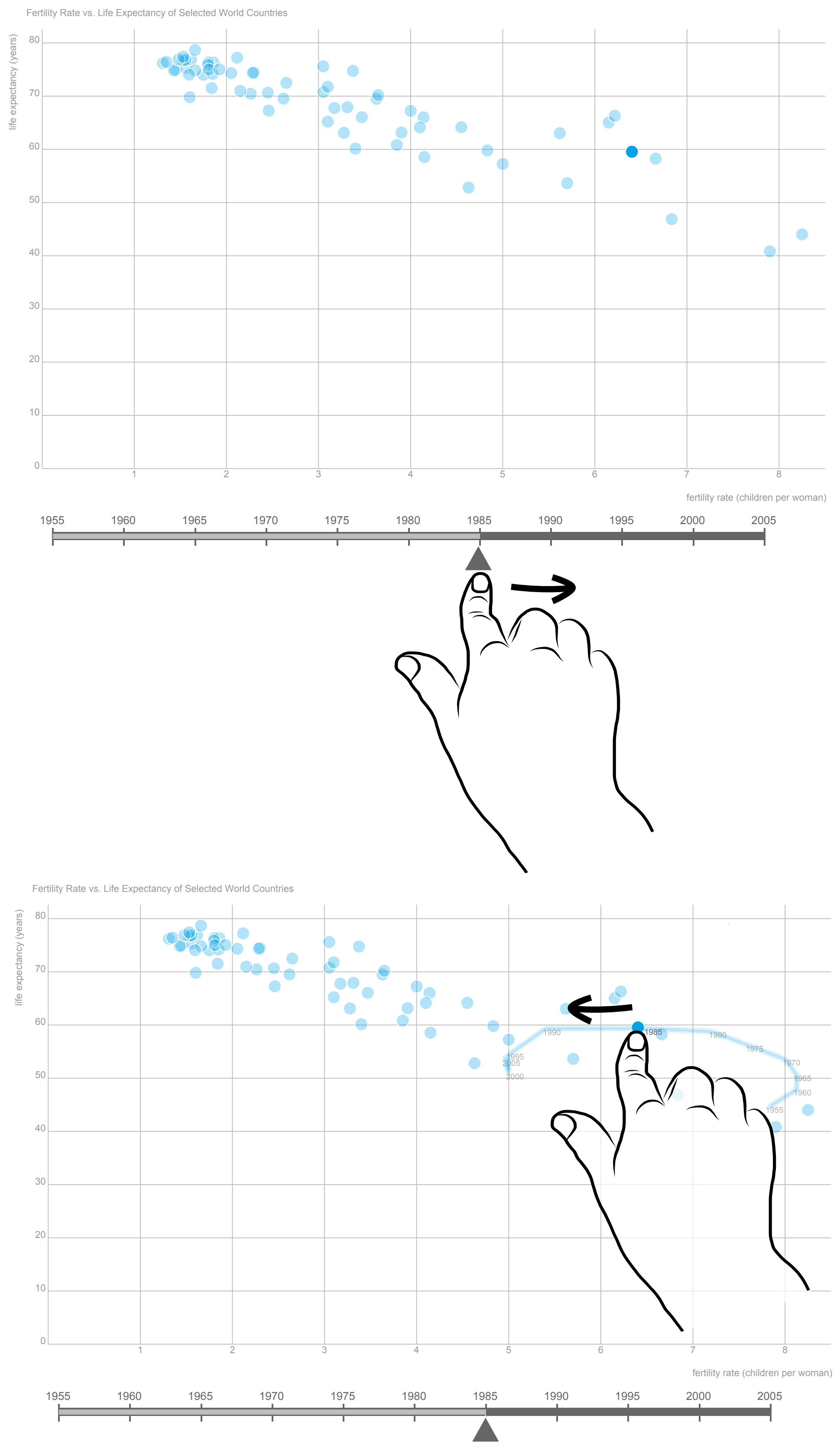

Fig. 1.2 Temporal navigation using a time slider (top) and object-centric temporal navigation invoked directly on a point in the scatter plot (bottom).

1.1 Motivation

Changes in data values over time are most often shown through animation, where the speed of animation is controlled by dragging along a time slider widget (Figure 1.2(top)). Alternatively, images of the visualization at each moment in time can be presented side-by-side (known as small multiples [55]). However, images do not convey motion, which is important for investigating and understanding temporal trends. Observing changes of individual data items using separate navigation controls, such as the time slider, can be cumbersome, mainly due to two related factors: visual instability and divided attention.

As more changes occur, consecutive views of the visualization become less similar and navigation becomes more visually unstable. Changes may involve only one data attribute (e.g., height of bars in a bar chart), or several attributes (e.g., element addition and removal, as well as node degree in a dynamic graph). In particular, navigating animated graphs is inaccurate for observing and tracing evolving local graph structures, mainly due to several simultaneous topological changes, causing layout instability [4]. For instance, Archambault et al. acknowledge that error rates found in their own and other related experiments evaluating the readability of dynamic graphs using animation and small multiples tend to be consistently high [4]. This suggests that while users can answer analytical questions reasonably fast (with some techniques) they are inadequately acquiring the required change information from the visualization to correctly answer those questions.

Sometimes, temporal navigation techniques are paired with visual highlighting to emphasize temporal changes, such as trajectory visualizations [46] or difference highlighting [47]. However the navigation and highlighting techniques are disjoint, requiring the user to shift their focus between them, when navigating time.

Since the time slider is separated from the data items, tracking and understanding how items change over time requires divided attention — manipulating the slider while observing how items of interest change. Dragicevic et al. created a non-linear video browsing technique, where any visual object can be dragged along its motion trajectory to navigate time [22]. Likewise, Wolter et al. designed a technique for dragging visualization objects along their 3D motion trajectories for navigating scientific visualizations [57]. While residing in different domains, both techniques were designed to solve a similar problem: the time slider is unsuitable for answering questions targeting the visual space, such as “Find the moment in the video when the car starts moving” [22] precisely, mainly because it is difficult to focus on the changes of individual visual objects.

This work focuses on designing interaction techniques that invoke object-centric temporal navigation. In our approach, the changing data item is used as the control for temporal navigation, enabling simultaneous exploration of the time dimension while focusing on the changing data item of interest (Figure 1.2(bottom)). To guide navigation, subtle visual aids for portraying the temporal trend of a data item are revealed during interaction. Object-centric navigation aims to provide a more intuitive and engaging technique for tracking, interpreting and exploring the changes of individual data items, by reinforcing focus on the changing item of interest. Contrary to most existing navigation techniques, our approach enables temporal navigation directly in the visual space.

1.2 Contribution

The main contributions of this work are:

Object-centric temporal navigation An interaction technique for exploring varying information visualizations and intuitively answering spatial queries, by directly interacting with data items of interest.

DimpVis An object-centric navigation technique for querying and exploring changes of data items in time-varying charts. We demonstrated how DimpVis can be designed to manipulate different temporally evolving visual variables (position, size, colour), in a variety of common chart types (scatter plot, bar chart, pie chart, heat map).

Glidgets An interface for exploring low-level, topological changes in dynamic graphs using interactive glyphs visualizing node degree and node and edge persistence changes over time. Object-centric temporal navigation is performed directly on a time slider embedded in the glyphs.

1.3 Organization

Chapter 2 presents a review of work related to existing temporal navigation and object-centric interaction techniques. Chapter 3 discusses the design of DimpVis and presents a use case scenario. Chapter 4 describes and presents results from an evaluation of DimpVis. Chapter 5 discusses the design of Glidgets and presents a use case scenario. Chapter 6 describes and presents results from an evaluation of Glidgets. Lastly, Chapter 7 addresses some technique limitations and ideas for future work.

Chapter 2

Background

We review four main topics closely related to our work: *direct manipulation interaction styles* (Section 2.1), existing techniques that are similar to our *object-centric navigation technique* (Section 2.2), techniques for representing *temporal changes of items in visualizations* (Section 2.3) and *dynamic graphs* (Section 2.4), and lastly, *temporal navigation techniques for information visualizations* (Section 2.5) and *dynamic graphs* (Section 2.6).

2.1 Direct Manipulation Interfaces

In direct manipulation interfaces, the visual objects of interest are represented consistently, physical actions are simple and support continuous flow of interaction, and immediate visual feedback is provided in response to physical actions [53]. For example, dragging a slider to navigate time, while the visualization updates in real-time. Interaction techniques designed around these principles follow a user-centered model, providing capabilities to express intentions and manipulate objects to perform actions. The techniques can be analogous to real world manipulation, exhibiting implicit familiarity [53].

Instrumental interaction exploits the natural and direct use of instruments for manipulating objects, conforming to the metaphor of physical manipulation. The instrumental interaction model focuses on understanding the significance, utility and properties of the interaction instruments [9]. Beaudouin-Lafon defines interaction instruments as mediators between a user and an object of interest [9]. A good instrument is one with low indirectness, which integrates many actions with the input device, and provides highly similar mapping from physical actions to the object's response. For example, in interactive information visualizations, navigation instruments, such as zoomable viewports are often provided. To reduce the *temporal offset* between the instrument's action and the object's response, immediate visual feedback is preferred.

Newer developments in technology (e.g., input devices) and data characteristics (e.g., complex size and structure) instill less traditional interaction styles. POST-WIMP interfaces aim to improve the constraining aspects of WIMP (Windows, Icons, Menus, Pointers) interfaces, where elements of interest are typically indirectly maneuvered using intermediary widgets [9]. Such control widgets present barriers between the user and the data, reducing interface transparency [35].

Beyond the real-time feedback and feeling of engagement provided by direct manipulation interaction instruments (control widgets), direct interaction can be initiated on the data items. Existing interaction design approaches highlight the necessity for minimizing the distance between the interaction source and the target object (e.g. [9, 35]). Furthermore, several visualization-related interaction models have been derived from direct manipulation principles (e.g. [9, 24]). In the visual analytics domain, Endert et al. encourage direct manipulation of the visual representations of model outputs to adjust underlying model parameters, differing from the traditional method of using control panels [25].

2.2 Object-Centric Direct Manipulation

The principles of direct manipulation and recommendations for closer interaction between the visual object of interest and the user, encourage object-centric interaction techniques. In this work, we focus on direct manipulation of a single visual object as a method for temporal navigation, facilitating visual exploration.

Our DimpVis technique was inspired by DimP, an interface for non-linear video browsing initiated by “relative flow dragging,” an interaction technique for dragging objects in a video scene along their motion trajectories [22]. The DRAGON interface uses a similar approach for in-scene video scrubbing [32]. For temporal navigation of 3D scientific visualizations, Wolter et al. created a system where visualization objects can be dragged along their motion trajectories, invoking corresponding forward or backward movement in time [57].

The Design-by-Dragging interface is composed of “as-direct-as-possible” techniques (e.g. dragging along a visualized model) to explore effects of changing simulation inputs and outputs, and generate design alternatives [16]. DirectPaint merges the space and time controls for video animation authoring by using the visual element’s motion trajectory as a basis for direct space-time manipulation [50].

Direct manipulation in the value domain was introduced by Perin et al. [40] to query and navigate time-varying ranking tables. The Drag-Cell technique is used to scan through the values of a data table cell over time by dragging on top of the cell. The entire table is updated when the dragging ends. The Vis-Rank technique reveals a transient line chart to visualize the values of a table entry and directly navigate to time points. While closely related to our DimpVis technique, in that direct interaction invoked on the target object is used for navigating time, an important distinction is that DimpVis leverages "embodied interaction" [21] and a high degree of interaction compatibility [9]. Rather than adjusting abstract numbers, in DimpVis the finger or pointer remains connected to the data item as it is manipulated through the spatio-temporal value domain and the item's temporal trend is revealed during dragging to guide navigation.

Moscovich et al. introduced a topology-aware navigation technique for graph visualizations, in which a slider is dragged along an edge to navigate to distant nodes [37]. While not applied to time-varying graphs, this technique exploits an object-centric interaction style similar to ours. In addition to exploration, direct manipulation can be used for data editing. Baudel presented a framework for directly manipulating data attributes, exemplified in the scatter plot view where point values are adjusted by dragging them [7]. Similarly, techniques in the form of dragging visual objects have been deployed in visual analytic systems as a means for altering the underlying model parameters (e.g. [13, 26]). This reduces the cognitive demand of learning complex models and their parameters, offering an intuitive method for model-steering [25].

2.3 Trajectory Visualization

One common way to reveal temporal trends in dynamic data is to use a trace visualization, which explicitly overlays a data object's changing values at various time points, onto the same visualization. This technique has been used to illustrate trends of points in time-varying scatter plots. The popular Gapminder Trendalyzer system uses animation and traces which dynamically draws a point's position on the graph, showing its temporal evolution [46]. Trajectories of points in scatter plots have also been used in visual analytic domains, such as analyzing time-varying financial (e.g. [52]), or medical datasets (e.g. patient data [44]). Using a stacked-bands approach atop a map, Tominski et al. used trajectory visualization to analyze spatio-temporal data, including attributes about data points embedded in the trajectory [54].

In our DimpVis technique, we build on the idea of trajectories, by designing visual hint paths, or indicators of all visual states of an individual data object across time. However, instead of simply displaying the trajectory visualization, we enable temporal navigation along the trajectory.

2.4 Visualizing Temporal Changes in Graphs

Visualization techniques can be used to highlight changes of graph elements, presented separately, or directly on the graph elements. Difference highlights (*difference map* [5], *difference layer* [59]) are drawn around graph elements and colour is used to differentiate inserted, removed and unchanged elements. They can be combined with temporal navigation techniques such as animation [6] or small multiples [47] to accentuate topological changes. Difference highlights are useful for directly comparing the graph at two consecutive time slices. However, with non-consecutive time slices, the highlights show an aggregation of the changes that occurred, not the process of change over the time span. TempoVis partially resolves this by showing the emergence of nodes and edges using colour, while varying the intensity to encode element aging [2]. This way, elements that emerged prior to the current time point, appear fainter in the view.

Metrics used to describe the changes of individual elements (e.g., node degree) or the entire graph (e.g., number of nodes) can be visualized in separate charts, such as line charts [42, 49], or bar charts [2]. Saraiya et al. evaluated multiple designs for visualizing temporal changes of node attributes in dynamic graphs [51]. Specifically, they compared encoding attributes in the node's colour per time slice and navigating with a time slider, to showing all attributes over time embedded as a line chart in nodes. The results from their experiments indicate that overlaying data per time slice was better for analyzing the entire graph at a single time slice, whereas overlaying all values over time embedded in node was more effective for detecting outliers in a node's changing attributes. Building on this work, the glyphs provided by our Glidgets technique can be interactively revealed, for detailed analysis of elements, or hidden, for global analysis.

GraphFlow uses a static, flow-based visualization technique to summarize high-level graph changes through changes in graph metrics, making irregular or drastic temporal changes pre-attentive [18]. Other static representations use different layouts to emphasize the temporal dimension, such as vertically arranging graph nodes connected by lines, at each time slice using a parallel coordinates layout [14], or showing connectivity in dynamic ego networks using a tree ring layout [27].

Federico et al. address the problem of tracking nodes in social network analysis by visualizing the node trajectories, making nodes easier to track and locate in any given time slice [28]. Additionally, the trajectories can encode changing social network analysis metrics, such as centrality. In a 2.5D layout, the temporal trajectories are conveniently mapped to the third dimension, reducing visual clutter and enhancing readability. However, they become less practical when applied to 2D layouts, such as small multiples.

Our Glidgets technique extends previous approaches by embedding node and edge-specific glyphs into visual elements. These glyphs can be used for invoking object-centric temporal navigation and to view the dynamics of element properties over time.

2.5 Temporal Navigation in Information Visualization

Existing techniques for exploring time-varying information visualizations include of variations of methods such as filters, animation (with control widgets), and series of static images.

Temporal filters, often employed as separate widgets (e.g. [19]), can isolate or aggregate views within ranges of time. However, since all views are not visible at the same time it is difficult to observe how the visualization changes over time. The small multiples technique displays images of the visualization at each time slice, ordered by time, in a matrix layout [55]. While this technique separates all time steps for easy viewing, reading values, and comparison, its effectiveness degrades as the time line and dataset size increase.

Animation techniques present each snapshot of the visualization, one after the other. Smooth transitions can be used to ease or highlight the changes between time slices (Kriglstein et al. have contributed a survey [34]). However, tracking individual data objects can become cumbersome if too many objects are changing or when the visualization is highly cluttered, causing distraction (e.g. in animated scatter plots [45] and dynamic graphs [27]). Interactively exploring the time dimension with animation typically involves an indirect slider or a set of control buttons. Existing slider techniques can facilitate global exploration, however the navigation control is decoupled from the changing visual elements, requiring shifting attention between the widget and the visualization [9].

While not applied to time-varying data, Scatterdice, a multi-dimensional visual exploration tool for facilitating analysis and navigation of data in scatter plots, is relevant to our work as it provides direct control of the view transition animation by interacting with the visual space, as opposed to a time slider [23]. Likewise, our approach provides direct control over the speed of animation by using the target object as the control and the main focus during interaction.

Lenses can be used to explore enclosed and constrained regions of visual elements by showing visual representation alterations and exploring hypothetical visual states. Chronicle uses a temporal lens that records the creation of a graphical document and workflow histories [31]. Sliding the lens within the document permits dynamic surveying of different regions and direct navigation through time. Zhao et al. introduced Chronolenses, which facilitate exploratory and analytical tasks with time series data, visually filtering regions of interest with a lens and coupling analytical operations with direct manipulation techniques [60]. While powerful, lenses present subtle barriers to interaction by their constrained spatial extent, and they are generally used to isolate and temporally explore one area of a visualization while maintaining the global context. In contrast, our direct temporal navigation techniques are provided on all visual elements, and the changes that occur are globally applied.

2.6 Temporal Navigation in Dynamic Graphs

Dynamic graphs visualize temporally evolving datasets containing a set of entities that are connected by some relationship. Navigating dynamic graphs can pose several interaction challenges, primarily caused by layout instability due to many simultaneous element changes.

As discussed in Section 2.5, the small multiples technique is the simplest method for navigating dynamic graphs. However, to understand the progression of topological changes, such as cluster formation, images of the graph must be compared manually. This requires the viewer to remember changes from image-to-image. Alternatively, time can be mapped to a third dimension in space [11]. However, as with many 3D visualization techniques, issues can arise with readability (e.g., occlusion) and interactivity (e.g., 3D manipulation).

Animation clearly illustrates temporal changes using motion. However, it can still be cognitively demanding to remember and compare the temporal changes of graph elements, without additional effects such as coloured highlights, smooth transition effects, or algorithms that prioritize graph layout stability.

Various layout algorithms have been created to alleviate the negative effect of changing graph elements between time slices, such as by restricting node movement [48]. This is known as "preserving the mental map" [17]. However previous studies evaluating the effect of mental map preservation on dynamic graph readability suggest that preserving the mental map may not have significant advantages for improving graph comprehension [4, 48].

DiffAni, a hybrid graph analysis tool, allows an analyst to freely combine small multiples, animation and difference highlight representations of the graph as tiles laid out in a timeline, for temporal navigation [47]. By combining multiple techniques, the analyst can select the desired combination of representations, hopefully exploiting the advantages of each, for more flexible analysis of dynamic graphs. Another approach is to enhance the animation effect used to transition between time slices. GraphDiaries uses staged animated transitions to separate temporal changes occurring in the graph into three phases: element removal, layout/element transformation, and element addition [6]. Coloured halos are drawn around changing elements to denote the different stages of animation and user-controlled temporal navigation can be performed at different levels of detail (e.g., within the staged-animation or across multiple time steps).

Existing temporal navigation techniques are effective for examining high and low-level topology changes between neighbouring or distant time slices. However examining all low-level changes occurring to an individual element (e.g., when the node disappears and re-appears), mainly relies on the user's memory. Moreover, visual effects highlighting the changes tend to only temporarily draw attention to a change [6]. Our Glidgets technique reveals the patterns of change for individual elements through the entire time series.

Chapter 3

DimpVis

DimpVis (*DimP* [22] for information visualizations), is an object-centric navigation technique for querying and exploring the time dimension by interacting directly with individual data items. We demonstrate how DimpVis can be used to invoke temporal navigation by directly manipulating different visual variables (position, size and colour). In this chapter, we present the design of DimpVis and a scenario using DimpVis for the bar chart to explore real data.

3.1 Design

DimpVis allows object-centric exploration of an individual data item's temporally evolving visual variables through direct interaction with a time slider embedded in the item. By eliciting *integrated interaction*, DimpVis can engage the user and produce a "hands on" data experience [8]. The DimpVis design extends previous work [22, 32, 57] by applying the direct temporal navigation technique to 2D time-varying information visualizations.

The DimpVis design was guided by the following design goals:

D1 Object-centric Navigation Temporal navigation occurs along the data trajectory, instead of the time trajectory.

D2 Navigation Flexibility Provide both controlled temporal navigation, as well as accelerated navigation shortcuts.

D3 Directness Direct connection to a data object of interest is maintained during navigation; the navigation control is embedded in the object.

D4 Interaction Consistency Navigation requires only a single finger or mouse pointer, without complex gestures or modes.

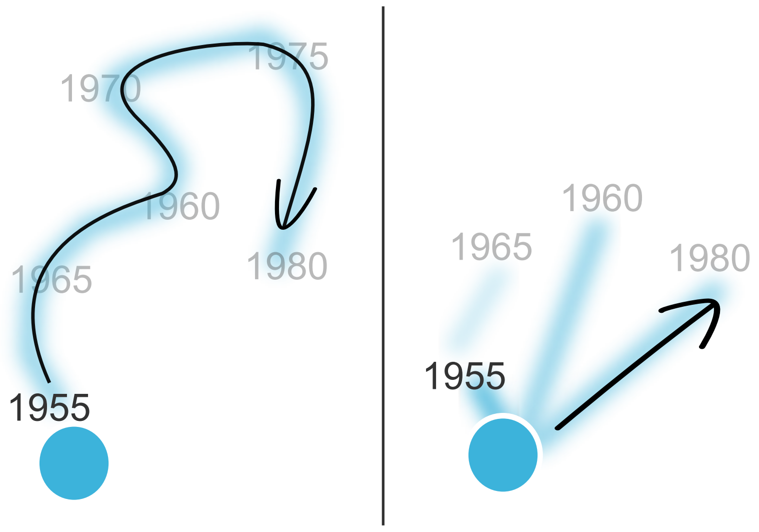

Fig. 3.1 The *time line* hint path follows temporal sequences (left), and the *flashlight* hint path connects to spatially adjacent time points (right).

D5 Minimal Visual Change DimpVis can be added to existing visualizations without changing the underlying visual representation. Visual additions should be minimally distracting and removed when not in use.

DimpVis consists of two main components: *hint paths* [22], visualizing how a data item changes over time, and *object-centric temporal navigation*, involving manipulating an item along its hint path. For clarity, the design components in this section are first discussed in terms of designing DimpVis for scatter plots. All of our web-based visualization prototypes were implemented using the D3 toolkit [10]. Section 3.2 discusses how DimpVis was designed for manipulating size, in pie charts and bar charts. Lastly, Section 3.3 discusses how DimpVis was designed for manipulating visual variables with no spatial motion, namely coloured cells in a heat map.

As a result of our evaluation (Chapter 4), we designed and implemented two different types of hint paths for the scatter plot and bar chart.

3.1.1 Hint Path Design

A trajectory is an aggregated representation of all changes of a data object. In animated visualizations, trajectories are useful for trend analysis and pattern detection of time-varying data [34]. Visual feedback, or *hint paths* [22] can help guide interaction. Therefore the hint path for the active data object (point in a scatter plot) is displayed to guide the interaction and provide contextual awareness during navigation.

To form a point's hint path, position is mapped to time. Following our design guidelines, the hint path should present the temporal evolution of a point in a clear, easily interpretable way, while also guiding fast and flexible temporal navigation (D1, D2). When not in use, hint paths are hidden (D5). We explored two design alternatives for the hint path: *time line* and *flashlight*.

Time line hint paths (Figure 3.1(left)): The positions of a point are linearly joined to form a path, ordered by time. Viewing this hint path reveals the patterns of change for a point over time. This design favours temporal trend legibility and navigation along the data trajectory (D1). It enables a user to trace the movement of a point through time, engendering a feeling for the data sequence through direct manipulation (D3). Navigation is linear in time, analogous to moving along a traditional time line. Interaction flexibility (D2) is sacrificed in that navigation is constrained to the temporal order of the path, reducing the speed of long-distance temporal navigation.

Flashlight hint paths (Figure 3.1(right)): Similar to preview bubbles [16], the closest positions of the point are dynamically revealed as the navigation progresses. As the point is dragged, the positions nearest to the dragging direction, where the point exists at any time, are shown, regardless of temporal order. This design favours speed and flexibility of temporal interaction (D2). It enables fast navigation to moments in time where the point has a certain position (value) (D2, D3). However, since positions of the point are not connected in temporal order, the temporal trend is not apparent. While the flashlight supports exploration of data point positions in any temporal order, it violates D1.

Each hint path design offers advantages targeted at different analyst intentions. The time line is designed for understanding the temporal trend of a point, while the flashlight supports direct accesss to a time when a point exists at a certain position. We selected the time line design for our comparative evaluation (Chapter 4), because it clearly illustrates a point's temporal trend, making its layout similar to a time slider. Navigation using the flashlight is more of a direct spatial query, as opposed to simulating a time slider embedded in the point. To overcome the main limitation of the time line design and support D2, a "fast-forwarding" feature was added to the path (Section 3.1.3), for quickly jumping to distant times.

Labels are added along the path to mark the point's position at each time interval [32], to show temporal location. Following D5, the hint paths should be subtle, therefore we blur them and use faint colours. Hint paths are only revealed for selected objects, and disappear when interaction ends.

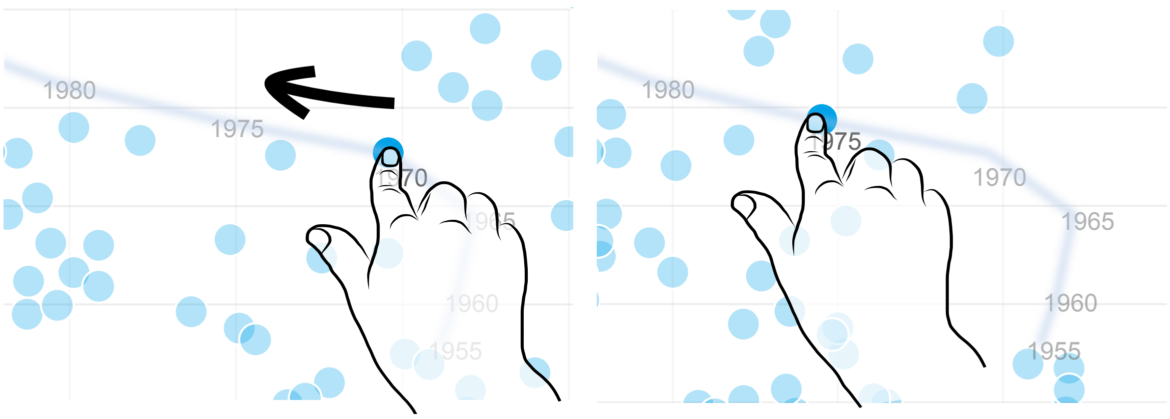

Fig. 3.2 Navigation in time for scatter plots is achieved by dragging a selected point along its hint path.

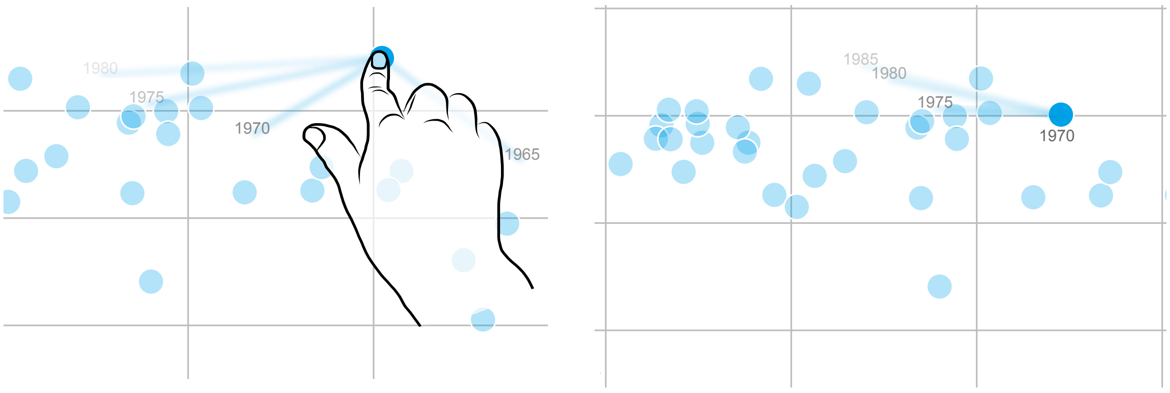

Fig. 3.3 Using the flashlight hint path, any point can be freely dragged (left), and the path reveals the closest positions. When a point is released, it snaps to the nearest position on the path, and the scatter plot is updated to that time (right).

3.1.2 Object-Centric Temporal Navigation with Dragging

Object-centric navigation allows the user to explore how a point changes over time, while remaining focused on it. To navigate time using a time line hint path, a point is dragged along the path, and the rate of dragging controls the speed of temporal navigation (Figure 3.2). Dragging has a high degree of compatibility, since the target object closely follows the action, and it lowers separation from the object of interest [9]. The position of the finger is projected onto the path according to the minimum-distance point. Using this method when loops or overlaps are encountered along the path, may cause the point to unexpectedly jump if the user does not closely follow the path [22]. The temporal direction is indicated by the time labels along the hint path. While a point is dragged, the global time of the visualization is updated accordingly, and all other points are updated to their new positions.

Using the flashlight hint path, as a point is dragged, the positions where the point exists over time that are nearest to the dragging position are revealed, regardless of temporal order. Line segments are drawn from the dragged point to the nearest positions, indicated by time labels (Figure 3.3(left)). Here, transparency encodes proximity: positions closer to the dragged point are darker. The transparency is adjusted as the point is dragged. The user can freely drag the point anywhere in the scatter plot. However when dragging stops, the point is automatically re-positioned to the nearest position displayed on the flashlight path, and the rest of the visualization is animated in time (Figure 3.3(right)). Since flashlight navigation does not follow temporal order, the rest of the points are not animated when a point is dragged. In other words, time is frozen.

Touch Input

In time-varying visualizations, depending on the rate and types of changes occurring in a data item's visual attributes, complex, curved trajectories may form. The mouse is a precise input device for target selection (pointing) tasks. However, it is unsuitable for precisely controlling and following a path of movement, such as in drawing tasks [30]. Additionally, a mouse presents a secondary barrier which separates the user from the data, decreasing the level of directness and transparency of the interface [35]. On the other hand, gestural interfaces reduce this separation, resulting in easier and more natural manipulation of data objects. Through directly touching and moving a data point along a trajectory, somatic feedback about the data values is received. Interaction techniques which engage people in a physical experience of connection with and direct manipulation of data facilitate the creation and communication of meaning through doing [21]. Therefore, the need for precise navigation and directness motivates us to prefer touch input for dragging a point along its hint path. However, mouse input is also supported.

Temporal Ambiguity

Temporal ambiguity has been recognized as a challenge for direct manipulation video browsing techniques [33]. Object-centric navigation along both types of hint paths becomes ambiguous when the point's position does not change across two or more consecutive time points. Additionally, in a flashlight hint path, ambiguity can occur when the point's position is the same at multiple time points (since temporal navigation is unordered). For flashlight hint paths, ambiguity resolution is left for future work. Below we

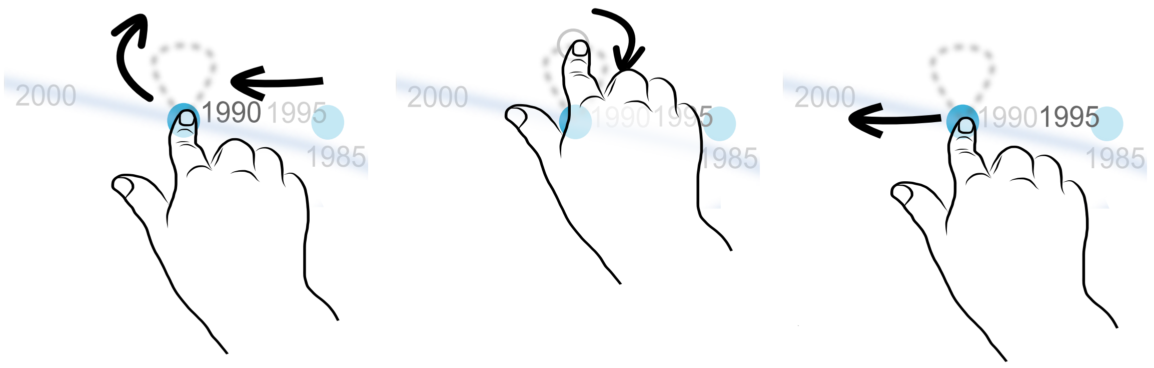

Fig. 3.4 Temporal ambiguity occurs in a scatter plot when a point does not move between time steps. Temporal navigation capabilities are provided in ambiguous using loops in which one transit around the loop corresponds to one time step.

discuss our ambiguity resolution for the time line hint path.

We use *interaction detours* integrated into the time line hint path, at areas of temporal ambiguity. The detours are designed to maintain the flow of dragging (D1) without a loss of directness (D3) or a need for disruptive or complex gestures, such as multi-touch interaction (D4).

When a point does not change position, we insert loops into the time line hint path [33]. When dragging around a loop, the point does not move, but an outline of it is dragged around the loop, to maintain connection between the finger and its temporal position on the hint path. To enter a loop, the user continues dragging in the current 2D direction of motion, moving in a continuous temporal direction (Figure 3.4). The dragging direction can be reversed inside the loop, to reverse the temporal direction. One full rotation around the loop moves forward or backward, by one time point. Labels are placed near the stationary point to show the time points covered by the loop. The current time point's label is highlighted.

Our first design assigned one loop to navigate all consecutive time points where the point's position became ambiguous; time points were spaced equally around the loop. While this design can accelerate temporal navigation, it is not visually scalable, since the loop's size expands according to the number of time points. In testing this design, we found it difficult to navigate time in detail, partially violating D2.

In areas of the time line hint path where the differences in point positions are very small, temporal navigation is challenging. In these regions, we also insert interaction detours. Therefore, in the scatter plot, loops are inserted where sequential points along the hint path are too close together.

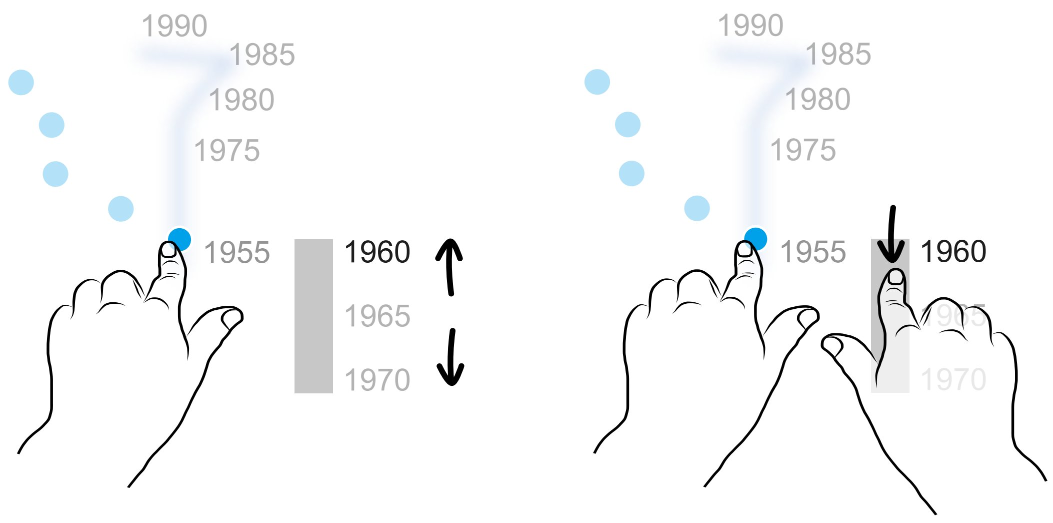

Fig. 3.5 In our multi-touch design, a second finger is used to control a vertical time slider to navigate through ambiguous regions.

During early prototyping, we also designed a technique using a second finger to activate and drag along a smaller, separate time slider (Figure 3.5). However this disconnects the finger from the point, and potentially requires more mental effort, violating D3 and D4. Alternatively, a "sticky motion" effect [33] could be used to skip through ambiguous regions. However this does not support detailed temporal navigation (D2).

Interaction Ambiguity

For time line hint paths, we define interaction ambiguity as points when the interaction (dragging) cannot be resolved to a unique temporal direction. Interaction ambiguity mainly occurs when cusps are formed along a hint path [22], where dragging in a certain direction can navigate in both directions in time. For example, if a scatter plot point's time line hint path doubles back on itself, at the point of reversal dragging along the hint path is temporally ambiguous.

Interaction ambiguity is resolved by maintaining *temporal continuity* at cusps. That is, we continue the navigation in the same *temporal direction* as it was moving prior to reaching the cusp (Figure 3.6). Consequently, temporal direction cannot be reversed at a cusp. When interaction starts at a cusp, there is no information for temporal continuity. In this case, forward time navigation is assumed. The direction can be reversed by changing dragging direction in a non-ambiguous area.

At some cusps, such as a sharp peak in the hint path of a point, our early testing showed

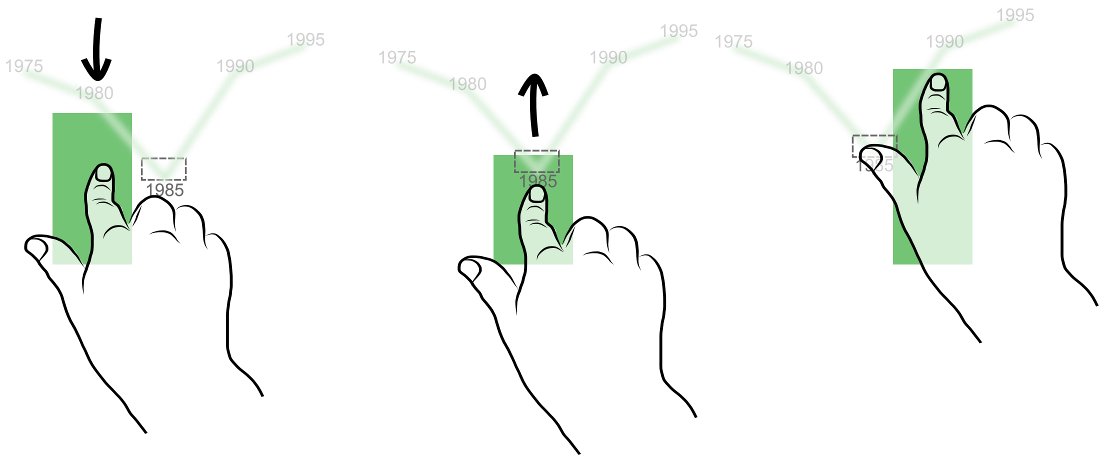

Fig. 3.6 At deep cusps, temporal direction is assumed to be forward. To continue navigation, a tolerance region (overlaid as a rectangle for illustrative purposes) detects changes in dragging direction. This way, the user need not drag all the way to the cusp.

it was cumbersome to bring the dragged object exactly to the cusp before reversing dragging direction to transit the point around the cusp. Frequently, users reversed dragging direction slightly before the point reached the cusp, leading to an unwanted reversal in the time direction. To ameliorate this problem, *tolerance regions* are applied to the cusp in which temporal continuity is enforced when the dragging direction changes near the peak (Figure 3.6). In this way, the cusp is 'rounded off' in interaction space and the user need not actually reach the peak in order to transit across it. The size of this region depends on sharpness of the peak; the tolerance region increases as the angle decreases.

To handle missing data values in the dataset, the path is interpolated using surrounding points. Exploring designs for differentiating missing data values from existing values on the hint path is beyond the scope of this work.

3.1.3 Additional Features

In order to support the design goals, several additional design elements are included in all implementations of DimpVis:

Time Slider: For high-level temporal navigation, DimpVis is paired with a traditional time slider widget. To move forward or backward in time, a small triangle tick is dragged horizontally.

Flexible Dragging: The finger can deviate away from the path during dragging, as though an elastic were connected to the nearest point. This is beneficial when using touch

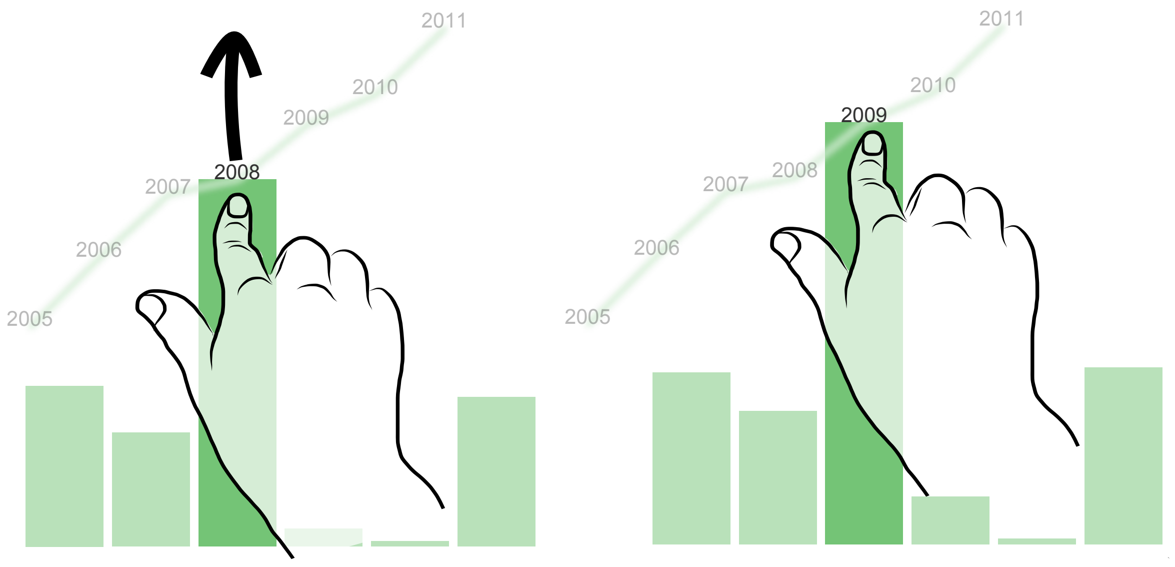

Fig. 3.7 Navigation in time for bar charts is achieved by dragging a selected bar vertically along its hint path. The hint path slides horizontally to stay connected with the bar and finger.

screens, where the hands may occlude the visualization.

Snapping to Time Points: As data values for a point exist only at labelled positions along the hint path, after a point is released from dragging, it is automatically re-positioned to the closest time position on the path.

Fast-forwarding: The hint path can facilitate both ordered navigation (dragging) and jumping across time. Fast-forwarding through time is invoked by tapping any time label on the path.

3.2 DimpVis for Manipulating Size

Similar to changing position in a scatter plot, dragging can be directly mapped to the motion of changing size. Below we describe DimpVis for manipluating changing height (bar chart) and angle (pie chart).

3.2.1 Bar Chart

Bar charts encode scalar data values in the height of bars, one for each data item or category.

Time Line Hint Path: All heights of a bar over time are connected by linearly inter-

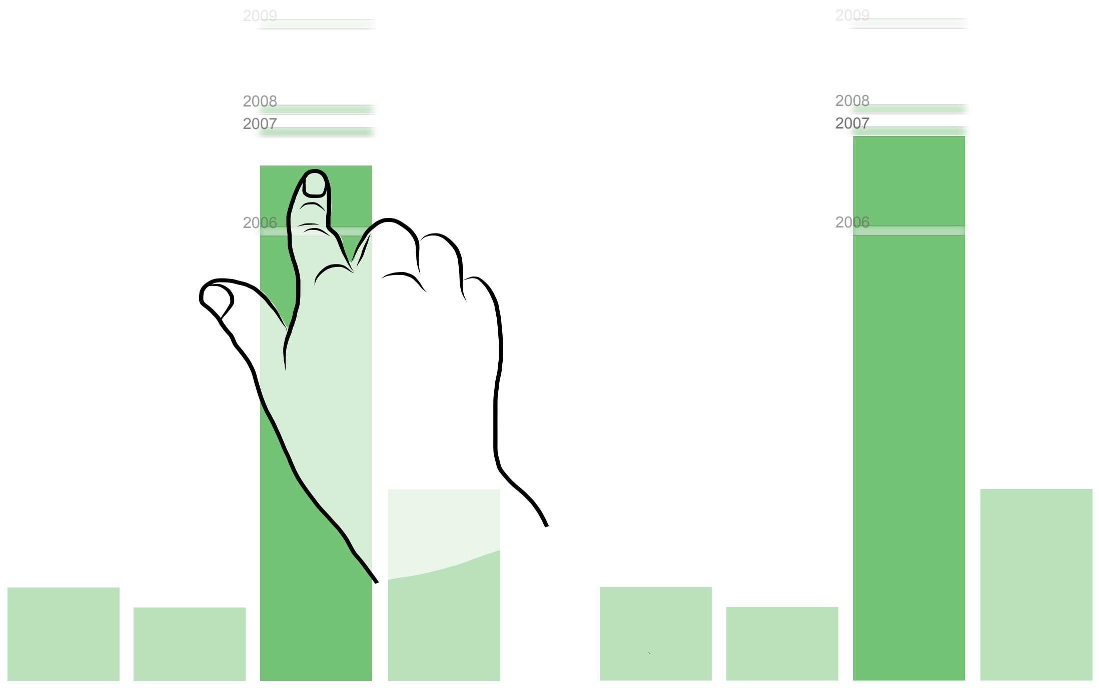

Fig. 3.8 The flashlight hint path for a bar chart. Any bar can be freely dragged (left), and the path reveals the closest heights. When a bar is released, it snaps to the nearest height on the path, and the bar chart is updated to that time (right).

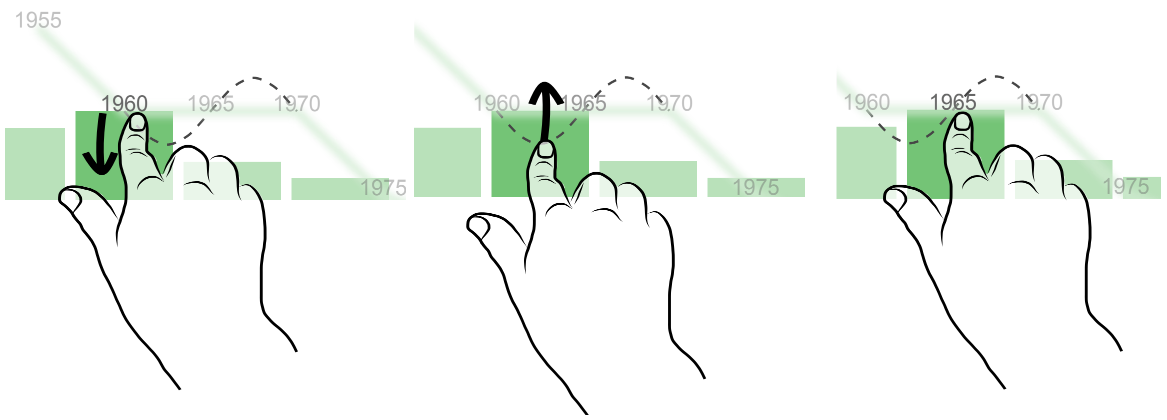

Fig. 3.9 Sine waves are introduced to provide an interaction technique to navigate through time periods where the bar height is not changing. The period of the wave is set so that the finger returns to the bar at each time step.

polated lines to show variations in height, forming a line chart of heights over time from left to right (Figure 3.7). As a bar is dragged vertically towards an adjacent height in time, the hint path translates horizontally . When the next time point is reached, its label is centered on the dragged bar.

Dragging Bars: To navigate time, a bar is dragged vertically, according to the time line path. Horizontal dragging could be used, since the hint path is presented as a horizontal time line. However, this would violate D3, as the finger would leave the bar. Vertical dragging corresponds directly to the changing value. So, the horizontal hint path translation is synchronized to the vertical dragging motion such that the finger and hint path always intersect at the top of the bar and the current time point.

Flashlight Hint Path: As a bar is freely dragged vertically, the heights where the bar exists over time that are nearest to the dragging height are revealed. Line segments are drawn at each of the nearest heights and time is indicated by labels (Figure 3.8(left)). Similar to the scatter plot, transparency of the line segments encodes proximity, and the bar chart is updated in time only when the dragging stops (Figure 3.8).

Temporal Ambiguity: To navigate time spans when the bar stays at the same height, sine waves are drawn on top of the time line hint path as dotted lines. The period of the wave is set so that the finger returns to the bar at each time step, maintaining directness (D3). For example, when encountering an upwards peak, the user must first drag up, then down, to move to the next time (Figure 3.9). When the apex of the peak is reached, the user is halfway between time points. Due to directional ambiguity at peaks, temporal continuity is enforced. Therefore, navigating along a sine wave maintains interaction consistency (D4), by using only vertical dragging motion.

Interaction Ambiguity: Temporal continuity and tolerance levels are used when cusps are formed on the time line hint path. Sine waves are inserted to ease navigation across very close heights along the path. Additionally, when the bar has a zero value, a short, faded, grey bar is used as a placeholder to initialize interaction.

3.2.2 Pie Chart

Pie charts are used to display parts of a whole, such as percentage information. Angular sizes of segments encode scalar values.

Time Line Hint Path: All angles of a pie chart segment over time are drawn and connected by angular paths (Figure 3.10). The angles of the hint path are placed outward on different radii: radius encodes time. When the segment is dragged, the path is animated in the radial direction. Using the chart's center as a reference, the path shrinks inwards

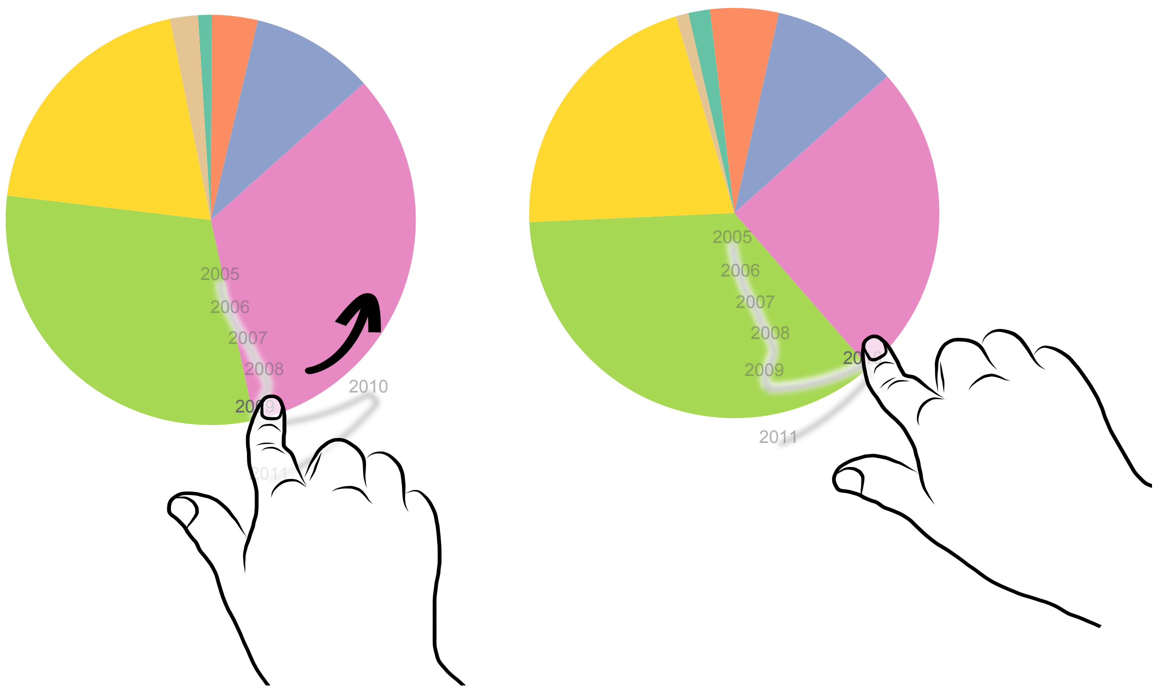

Fig. 3.10 Pie charts are adjusted in time by dragging an edge of a segment in angular directions along its hint path. Here, the purple segment is dragged and the hint path slides in and out along the radius to remain connected with the finger position and edge of the segment.

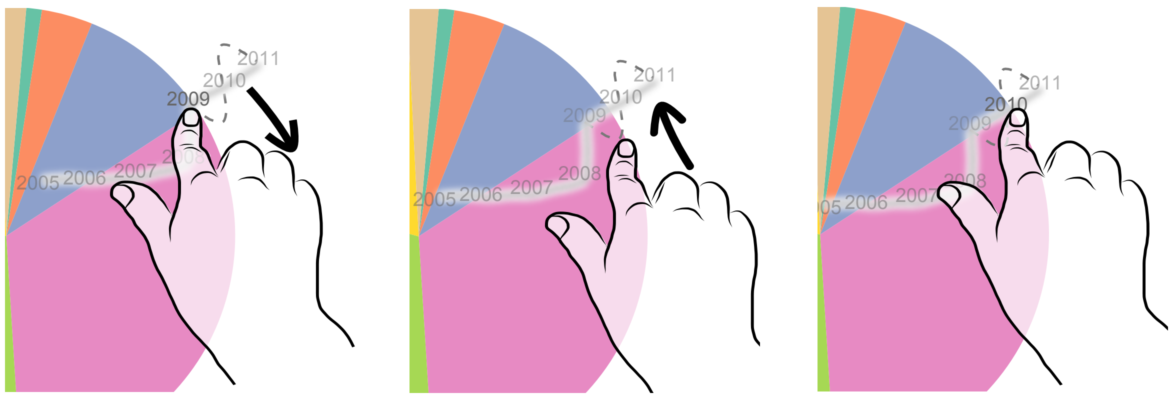

Fig. 3.11 Sine waves are added to pie chart hint paths when a segment doesn't change angle for some consecutive time points. Here, the blue segment is stationary for three years and repeated, angular dragging is used to navigate the wave.

when moving forward in time and expands outwards when moving backward. In our first design, radius was increased only when angular dragging direction changed. This way, it was possible for multiple angles over time to be drawn on the same radius. While this design is more space efficient, informal testing with colleagues showed that it was difficult to understand and confusing. Additionally, this design did not entirely conform to our time line hint path, since the time dimension was not assigned a unique visual attribute.

Dragging Segments: One side of the segment remains stationary, while the other side can be dragged to resize the segment's angle. Keeping one side stationary controls the dragging and makes the angles more readable. While a segment is dragged, all other segments are resized according to data for the updated time point. The radial hint path translation is synchronized to the angular dragging motion so that the finger and hint path always intersect at the edge of the segment and the current time point.

Temporal Ambiguity: Sine waves in the hint path are used as detours, using angular motion to navigate through them (Figure 3.11). One time step corresponds to half a period of the wave, so that the finger always returns to the dragged segment at each time point.

Interaction Ambiguity: Ambiguous interaction occurs whenever a cusp is formed on the hint path, indicating a change in angular dragging direction. Therefore, temporal continuity is enforced at cusps and tolerance regions are also applied.

3.3 DimpVis for Non-spatial Visual Variables

DimpVis provides a temporally ordered visual scan of all values of a data item. When the changing visual variable has spatial motion, dragging is conveniently mapped to the changing variable, creating a direct correspondence between the visual feedback and user interaction. However when there is no motion, such as changing colour over time, the correspondence must be reinforced by the hint path, to maintain connection between the finger and the data item. This interaction design is similar to the DRAG-CELL technique, where values over time are browsed by dragging in the value domain [40]. The design challenge here is finding an appropriate mapping between the interaction (dragging) and the non-spatial visual variable.

3.3.1 Heat Map

A heat map can visualize information using colour. In our design, the heat map is a correlation matrix plot, where information corresponding to the strength of connections corre-

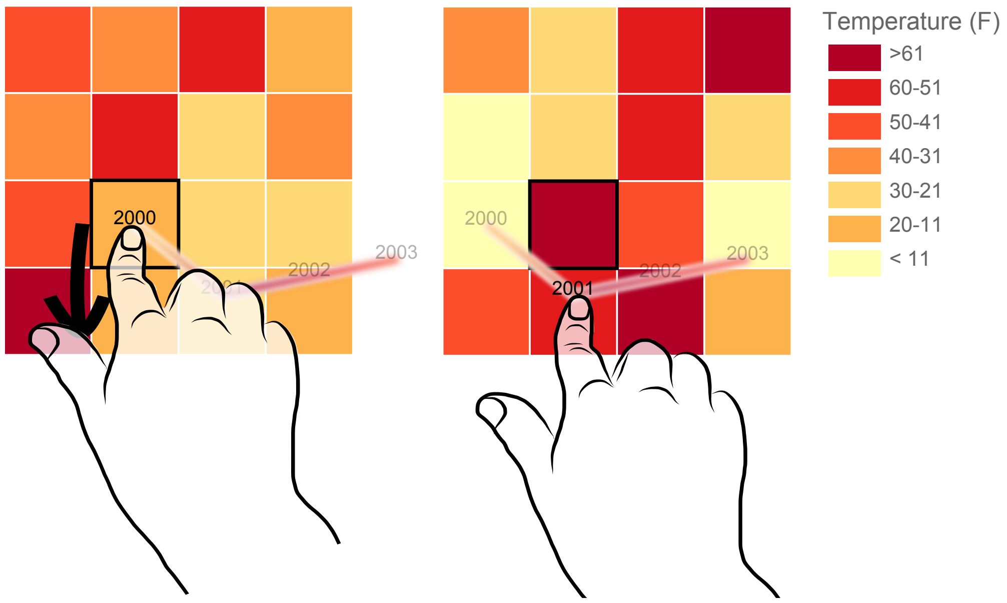

Fig. 3.12 Heat maps are adjusted in time by dragging vertically in the space of the colour scale. The hint path slides horizontally to stay connected with the cell and finger.

sponds to row/column intersections is encoded using colour. A time-varying correlation matrix could be used to study the changing strength of connections between friends in a social network.

Time Line Hint Path: Using the same design as the bar chart's hint path, the data values are plotted along a horizontal time line, where the y-position is relative to the corresponding vertical position on the colour scale (Figure 3.12). The hint path is also coloured to show the colour at each time point, and a gradient is used along the interpolated segments to show the transition. Therefore, the hint path not only illustrates variation in colour, but allows for relative comparisons between colours according to the numerical value they encode. In general, the challenge of this design is that colour does not intuitively associate with a quantitative amount, making both the hint path and interaction more abstract.

Dragging Coloured Cells: Dragging is initiated by touching a cell. As colour doesn't have an inherent spatialization, we use the arrangement on the colour scale to provide one, thus dragging occurs vertically for a vertical colour scale. As dragging occurs, the colour of all cells is linearly interpolated to represent the current point in time. The hint path translates horizontally to maintain the direct connection of the finger to the hint path while dragging in the data-space direction. Note that direct connection to the cell may be lost, violating D3, as the cell itself does not move with the finger.

Temporal and Interaction Ambiguity: Ambiguities in the heat map's hint path and interaction are handled in the same way as for the bar chart.

3.3.2 Public Demonstrations

Before evaluating DimpVis, our four prototypes were demonstrated at a public workshop (Surfnet 2013), to various colleagues in UOIT research labs and to researchers involved in the 2014 CHI conference. From these demonstrations, people offered feedback based on their experience using the prototypes. In particular, one person suggested considering an evaluation using a mouse, as opposed to touch interaction. Another person mentioned that it was sometimes difficult to acquire the data items for dragging (i.e., the fat finger problem) and that adding extra space surrounding the item to detect touches might alleviate this program. There was some confusion with using DimpVis for the pie chart. People were unsure which side of the segment could be dragged and, at times, it was unclear which segment was being dragged since all segments are re-sized during dragging. An indicator highlighting the dragged segment would help draw attention to it. Additionally, the side of the segment nearest to the finger could be used for dragging, while the other remains stationary. This would require two mirrored versions of the hint path, to appear at either edge of the segment. While most of these design suggestions were not addressed in our current design, we hope to incorporate them in future design iterations.

3.4 Scenario

In this scenario, we illustrate how DimpVis can be used to analyze a time-varying dataset using both the flashlight and time line hint path designs.

Sue is in charge of assessing the quality of programs offered at UOIT. As part of her assessment of program popularity, she would like to examine how the amount of students enrolled in programs has changed over the years. In particular, she wants to investigate and characterize temporal trends of enrollment for UOIT programs, examine the range of enrollment amounts for programs and look for anomalies in trends, such as a decrease in enrollment. These tasks are cumbersome using only the traditional time slider, because the temporal trend of a data item is not visually represented, and navigation is limited to manipulating the temporal dimension, separated from the visualization. To support her analysis, she uses DimpVis for a time-varying barchart, visualizing total enrollment for each program at UOIT, from 2005–2011 (http://cudo.cou.on.ca). Programs are grouped

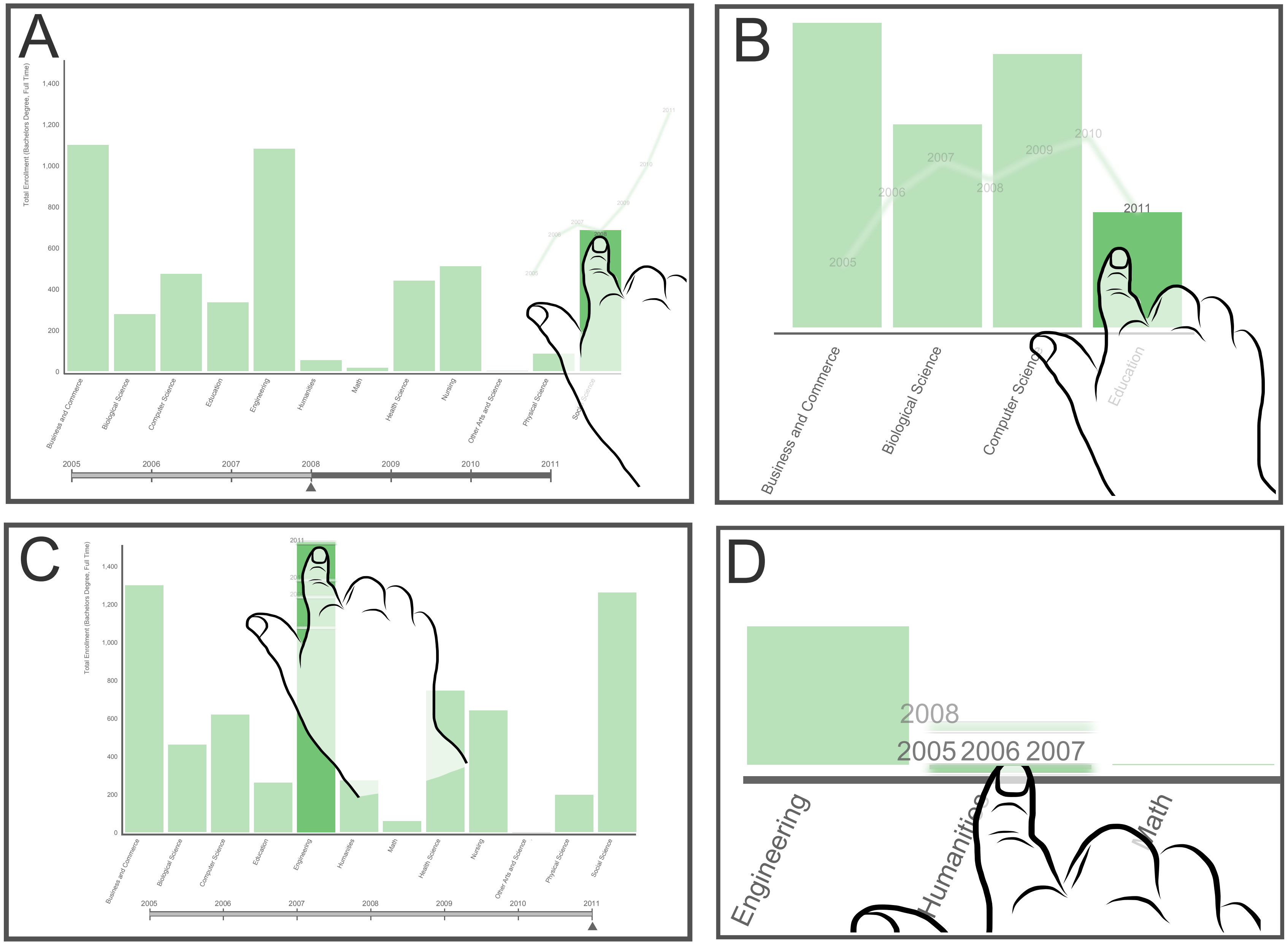

Fig. 3.13 Screenshots illustrating parts of the scenario: A) The decrease in enrollment of social science, B) Investigating the irregular trend of education, C) Using the flashlight path to query the highest enrollment of engineering, and D) Finding when humanities had zero students enrolled.

along the horizontal x-axis, and total enrollment is encoded by the height of bars.

To get an overview of how the entire dataset changes, Sue drags the horizontal time slider and sees that, as expected, the enrollment of most programs increases over time. She notices that the social science and education programs suddenly decrease, at some moment in time. She taps the social science bar to reveal its time line hint path. She can immediately see on the path that enrollment drops, once in 2008. She goes directly to that year by dragging the bar and following the hint path (Figure 5.9(A)). Then, at 2008, she taps the education bar and drags along its hint path. The path reveals that its enrollment has also decreased in 2008, and again in 2011 (Figure 5.9(B)). She decides that these two programs have followed an irregular pattern over time, and makes note to further investigate them.

In terms of popularity, engineering seems to have the highest enrollment overall. Sue switches to the flashlight hint path and drags engineering towards the top of the vertical axis and the path indicates that engineering's highest enrollment was in 2011, at 1600 students (Figure 5.9(C)). Sue also notices that the enrollment amount of the other arts and sciences and humanities remains low, relative to the other programs. She uses flashlight hint path to issue a direct query, to see if and when the enrollment of these programs reaches zero. She drags the humanities bar down to axis and sees that it reaches zero for the first three years, which leads her to believe that the program did not exist, until 2008 (Figure 5.9(D)). She then drags the "other" bar down to the axis and sees that it goes to zero for the last two years. Sue then decides to further investigate the programs that were assigned to the "other" program category.

Chapter 4

Evaluating DimpVis

Using the bar chart and scatter plot, we performed a quantitative, task-based evaluation comparing DimpVis to the traditional time slider and small multiples. In this chapter we discuss the design of our evaluation, the results and list some implications for the DimpVis design, based on our results.

4.1 Evaluation

We performed a comparative evaluation between three temporal navigation techniques: DimpVis, the traditional time slider, and small multiples, measuring their performance (time and error rate) when used to complete tasks involving reading values and observing trends of data objects. Additionally, we created a smaller set of extra tasks (not included in our analysis of performance measures) using the interaction detours (loops and waves). To keep experimental sessions a reasonable length of time, we decided to evaluate two visualization types (bar chart and scatter plot) from our four prototypes. Participants completed sets of tasks using each interaction technique, with both the bar chart and the scatter plot. We chose to evaluate the scatter plot and bar chart as it was thought that participants would be familiar with reading them (as opposed to reading a time-varying heat map).

4.1.1 Task and Dataset Design

To evaluate the performance of each interaction technique, we created a set of short analytical tasks. Our tasks targeted the visual space, as opposed to the temporal space, coinciding with related experiments [22, 57]. We characterize visual space tasks as finding when a certain data object has a specified visualized property, concentrating on observing the data object's value (e.g. "When is bar A at height 3?"). Conversely, temporal space tasks involve finding a specified data object's value given a moment in time, concentrating on navigating the time line (e.g. "At 1995, what is the height of bar A?").

We designed tasks which required a participant to observe and quantify a data object's individual value or, its trend of values changing over time. Additionally, we included tasks for comparing changing values of multiple data objects. While DimpVis is an object-centric technique intended to focus on changes of single data objects, comparison tasks were added to test the effect of divided attention between navigation and other changing objects, on technique performance. Our tasks were mainly derived from a taxonomy of low level analytical tasks [3]. We generated four types of tasks:

Retrieve Value (RV): Read the changing value of a data object.

Comparison (CO): Compare the changing values of two data objects

Characterize Distribution (CD): Identify characteristics of the overall trend of a data object's changing value

Outlier Detection (OD): Find when a data object's changing value deviates from the overall trend of all other data objects

RV and CO are value-reading task types, while CD and OD are trend analysis task types. We generated three objective and three practice versions of each task type, per visualization and interaction technique (See Appendix A for all tasks). Each task version's difficulty was assessed during pilot testing to ensure participants were able to comprehend and complete it. Originally, for the bar chart, the CD task was extended to target multiple data objects, requiring participants to observe the trends of two changing data objects simultaneously (e.g. "Find a moment when bar A and B change from increasing to decreasing"). However, during pilot testing this task was found to be frustratingly difficult by nearly all six participants, and was much more time consuming than the other tasks. Therefore, we eliminated this task from our evaluation. For the same reason, the OD task was not created for the bar chart because participants had difficulty observing simultaneous motion of bars.

To ensure the data exhibited realistic behaviour, we started with real datasets and made adjustments to ensure each task had a unique correct answer. The specific data objects and question details were varied across all interaction techniques, and the practice and objective tasks. In each task version, the target data object was always different. Therefore, a participant never encountered a task targeting the same data object, with the same values. The time pointer was set to the starting year at the beginning of each task. For all tasks, the correct solution (year) was placed somewhere in between the middle to the last year. This ensured temporal navigation was required to complete the task.

For the scatter plot, datasets always contained 20 points (axes artificially labelled as age and height), over 10 years, and datasets for the bar chart always contained 13 bars, over 10 years. Data labels were also artificially created as 2000–2009 (for years) and randomly assigned letters of the alphabet (for the data objects).

In summary, we used a 3 *technique* (DimpVis, slider and small multiples) x 2 *visualization* (scatter plot, bar chart) within subjects design. The order of *technique* and *visualization* were counterbalanced with two participants for each ordering, resulting in a total of 12 participants. For bar charts there were 3 *task types* (RV, CO, CD), and for scatter plots there were 4 *task types* (RV, CO, CD and OD). *Task type* ordering was randomized across participants. In total, there were *technique* (3) x *task type* (4) x *task versions* (3) x 12 participants= 432 trials for the scatter plot and *technique* (3) x *task type* (3) x *task versions* (3) x 12 participants= 324 trials for the bar chart. In addition, for each visualization type, participants completed three task versions for the RV and CD tasks (total 6 trials per visualization) on datasets with temporal ambiguities using DimpVis with interaction detours. These tasks trials were only used to inspire subjective feedback and not included in quanitative analysis.

4.1.2 Procedure

The following procedure was carried out twice for each participant, once for each visualization type. Half the participants started with scatter plot, and half with bar chart.

At the beginning of the session, participants watched a video explaining time-varying visualizations and a demonstration of the steps required to complete a task. Prior to using each interaction technique, participants were given an explanation of how to use it and a demonstration of how it works. The participant was instructed to complete each task as quickly and as accurately as possible. Participants were able to skip tasks, but they could not re-do them. However, in our results, no tasks were skipped. With each interaction technique, the participant first engaged in a set of practice tasks, followed by the objective tasks (used in our analysis). We ensured that participants never encountered an objective task they have not previously practiced by using the same amount and type of tasks in both the practice and objective task sets. The participant was not informed which tasks were for practicing.

At the start of each task, the participant was given as much time as needed to read the task description, which remained visible during task completion. When the participant pressed a “ready” button, the visualization was displayed. Data objects involved in tasks were highlighted in orange during the initial time step, allowing participants to pre-attentively locate them [47]. After passing the second time step, all data objects faded to the same colour, permanently. Solutions to the tasks were submitted as views of the visualization at a certain year, using the assigned interaction technique to navigate time.

For example, if using the DimpVis technique for the scatter plot, a participant would drag the point to a position which they thought answered the task. Task completion time was measured from when the “ready” button was pressed to when the answer was submitted (pressing a “submit” button). All completion times, submitted answers and user interactions were logged by the system. Error rate was measured by the amount of incorrect tasks out of the total number of task trials, using the logged answers. Participants were also video recorded, from over-the-shoulder. On screen feedback about correct and incorrect answers was provided after task submission.

After completing all objective tasks, participants were invited to rate each technique subjectively, using a 5-point Likert scale (1-Strongly Disagree to 5-Strongly Agree). Although the interaction detours are an extension of DimpVis, during pilot testing we found that participants had specific comments about them. Therefore, the interaction detours were rated separately.

Participants were then invited to use a full-featured version of the DimpVis technique in the bar chart and scatter plot prototypes, to explore real datasets. We provided some open-ended questions focusing on temporal trends of the data to inspire exploration (e.g., “Is there any common trend across all programs of how enrollment varies over time?”). Participants were instructed to freely explore the data by dragging the points or bars and while speaking aloud their analysis of the data.

Following the open exploration, participants completed a subjective feedback questionaire about the hint path. Then, the whole procedure was repeated for the second visualization type. Lastly, participants engaged in a short, semi-structured interview regarding their experience using DimpVis for both visualization types.

4.1.3 Interface Designs

We created three technique interfaces for each visualization type:

DimpVis: DimpVis with a restricted version of the time line hint path, revealing only the immediately adjacent time steps during dragging (Figure 4.1). A non-interactive time slider is included to show temporal location.

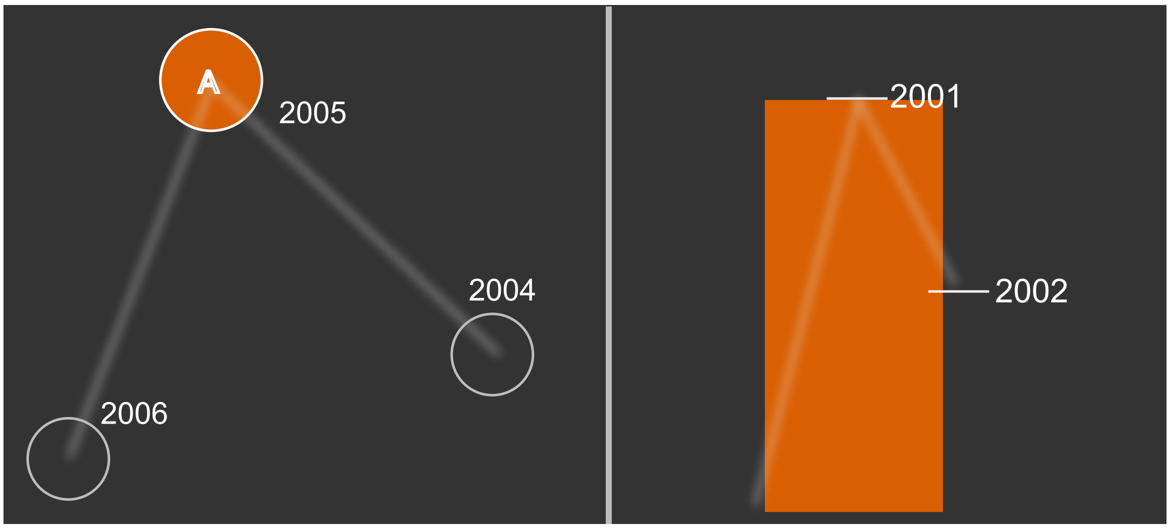

Fig. 4.1 The partial hint paths used in the scatter plot (left) and bar chart (right) evaluations.

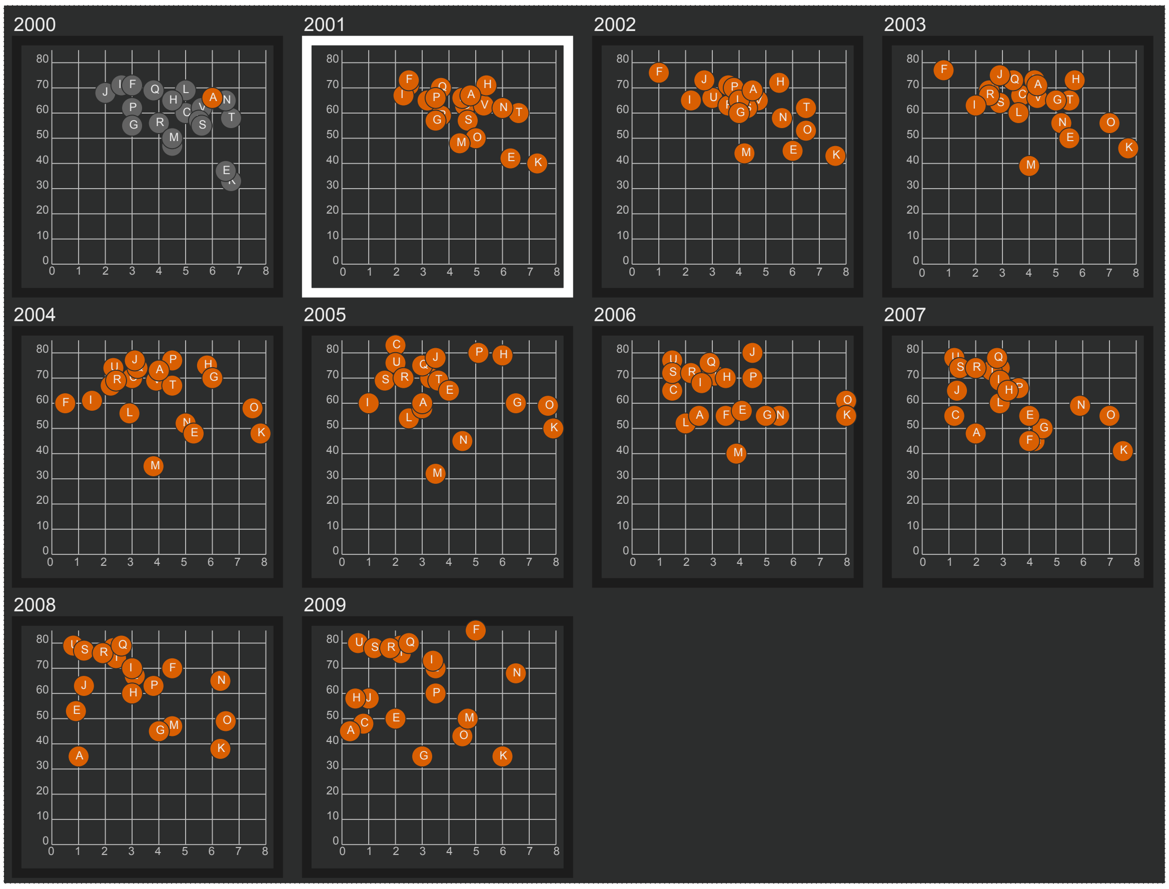

Fig. 4.2 The small multiples display used in the scatter plot evaluation. The white border shows an image that a participant has selected as their answer.

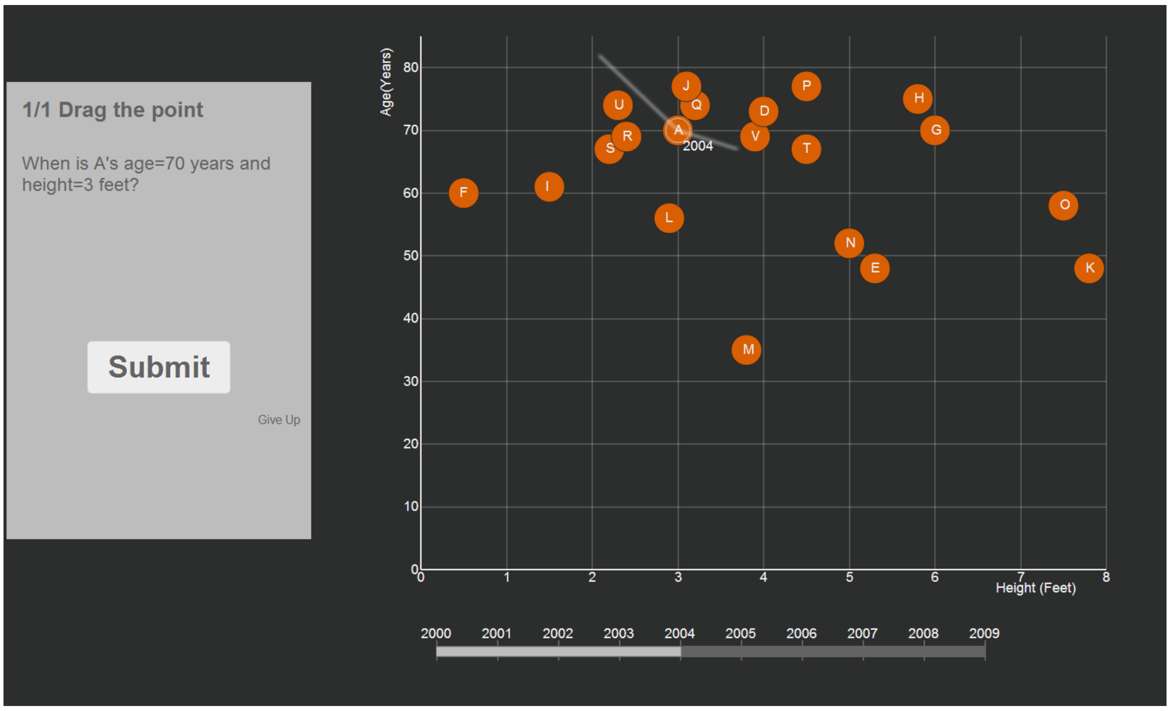

Fig. 4.3 The interface used during the scatter plot evaluation, consisting of the task description (left) and interactive scatter plot (right).

Time Slider: An interactive horizontal time slider bar is placed underneath the visualization, where years are shown at each tick mark.

Small Multiples: Equal-sized static images of the visualization at each year are displayed on the screen (Figure 4.2). The images are ordered by time, spanning from left to right, then top to bottom. Year labels are placed on top of the images. We ensured that data objects necessary to complete a task were clearly visible.

A constrained version of the DimpVis hint path was used for the experiment in order to focus on the feature of most interest — object-centric temporal navigation. This hint path indicated available dragging directions in the immediate area of interaction, but the whole hint path was not shown. This prevented simple reading of the complete hint path to answer task questions. No additional interaction capabilities (pan, zoom, filter, fast-forwarding) were provided. All colours were selected from a colour-blind friendly palette [12]. The task question was displayed in a left-hand sidebar, and the main part of the screen was used to show the visualization (Figure 4.3). Axes lines and data object labels were added to the visualizations.

While solutions to the tasks may be revealed on the restricted hint path, early pilot testing showed it was not practical to use DimpVis without a path, because the dragging

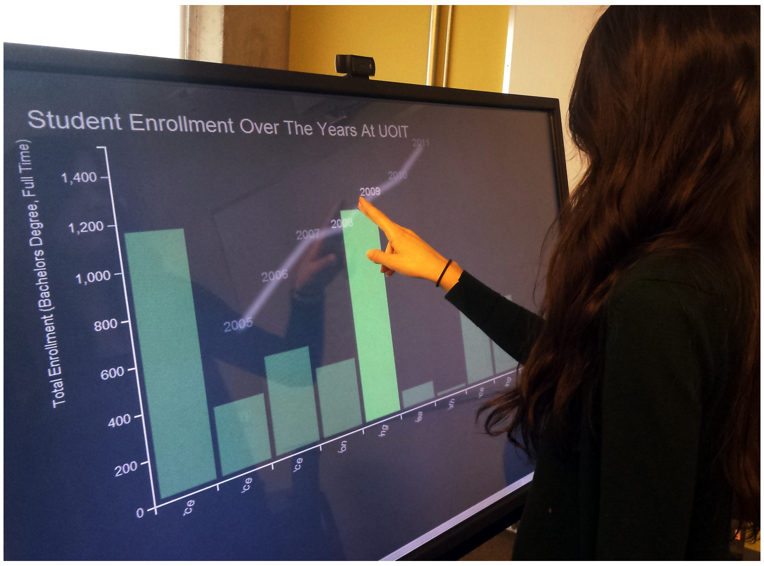
Student Enrollment Over The Years At UOIT
Total Enrollment (Bachelors Degree, Full Time)
1,400
1,200
1,000
800
600
400
200
0
2005
2006
2007
2008
2009
2010
2011

Fig. 4.4 In our evaluation, participants interacted with a large touch screen, while standing, to complete tasks.

direction is unknown. In any case, the effects of the hint path are an integrated part of the design of DimpVis, so it was included in the evaluation.

4.1.4 Pilot Test

Pilot testing helped refine the design of our experimental interfaces and assess the tasks. In the DimpVis experimental interface, one participant repeatedly attempted to drag the time slider despite being informed that the slider was only provided as a temporal indicator. Therefore, we removed the moveable tick used to interact with the slider and add time labels to the partial hint paths. Pilot testing also helped us improve the training materials. For instance, one participant suggested adding a clip to the tutorial video that demonstrates how a task is answered. Some tasks were modified, because the answer was not easily visible on the screen. For example, in one comparison task, the heights of two bars were too close, and not easily distinguishable. Additionally, grid lines were added to the charts, to help with reading data item values. The size of the loops was reduced because some participants commented that they were unnecessarily large and that they sometimes felt fatigued when dragging around them.

4.1.5 Experimental Setup

All experiment sessions were conducted in the same, private laboratory, with lower lighting conditions. A standard workstation computer and a wall-mounted 46-inch Phillips TV (1768x992 screen resolution) with a PQ Labs multi-touch overlay were used to run and display the visualization prototypes in a Google Chrome browser window. Participants used the touch screen to complete the tasks while standing (Figure 4.4). On average, the study lasted two hours. Participants were allowed to take breaks between tasks as needed, and received a gift card as compensation.

4.1.6 Participants

We recruited 13 participants (11 male and 2 female), aged between 19 and 30 years, from our university and surrounding area. One female participant's data was excluded from analysis due to an observed lack of effort to correctly complete the tasks, resulting in frequent incorrect answers and many skipped tasks. All remaining twelve participants self-declared as at least beginners in reading both bar charts and scatter plots, reading them for visual analysis at least a few times a year. All participants used touch screens daily (mainly phones or tablets), and the DimpVis technique was new to them.

4.1.7 Hypotheses

The tasks fall into two main categories: reading values of data objects (RV, CO) and observing the trend of a data object's changing values (CD, OD −scatter plot only). We argue that while DimpVis may not present a faster or more accurate method for reading values from visualizations, it may be more efficient for characterizing the trends of data objects. We suspected that the small multiples technique would perform better for reading values from a visualization, as opposed to observing trends, because motion is not apparent. Conversely, the slider and DimpVis might perform better for characterizing trends. Although sliders are effective for understanding global changes, tracking changes of individual data objects can be difficult due to the simultaneous motion of other data objects. DimpVis, however, reinforces focus on target data objects. Therefore, we hypothesize that, for both the bar chart and the scatter plot:

H1: Overall, DimpVis will be the fastest and most accurate for completing tasks, followed by the slider and then small multiples.

Table 4.1 RM-ANOVA results for scatter plot task completion times.

Factor	F-score	p-value
technique	$F_{1.33,14.65} = 11.725$	$p = 0.002$
task type	$F_{2.43,26.73} = 13.602$	$p < 0.001$
technique x task type	$F_{3.41,37.51} = 4.031$	$p = 0.011$

H2: Overall, tasks involving reading values (RV, then CO) will be faster and more accurate than trend-based tasks (CD, then OD).

H3: For reading values tasks (RV, CO), small multiples will be the fastest and most accurate, followed by the slider and then DimpVis.

H4: For trend-based tasks (CD, OD –scatter plot only), DimpVis will be the fastest and most accurate, followed by the slider and then small multiples.

4.2 Results

Below, we present quantitative (time and error rate) and qualitative (interview responses, subjective ratings and observations) results for the scatter plot and bar chart, as well as observations gathered during exploratory periods.

4.2.1 Scatter Plot: Quantitative Results

Task Completion Time

A two-way repeated measures ANOVA, with factors *technique* (3 levels) and *task type* (4 levels) was performed. The dependent variable, time (measured in seconds), was log-transformed to reduce skewing in the data caused by outliers. After transforming the data, Shapiro Wilk Tests indicated one variable was not normally distributed ($p < 0.05$), therefore one outlier was removed and replaced with the next value at least two standard deviations from the mean. Performing Shapiro Wilk tests again indicated that all variables were normally distributed ($p > 0.05$). Mauchly's test showed that the assumption of sphericity had been violated for main effects of *technique* ($\chi^2(2) = 6.972, p = 0.031, \epsilon = 0.67$) and *technique x task type* ($\chi^2(20) = 44.979, p = 0.002, \epsilon = 0.57$). Therefore, degrees of freedom for both effects were corrected using ϵ of the Greenhouse-Geisser estimates of sphericity. We adjusted significance values for all post hoc pairwise comparisons using the Bonferroni correction.

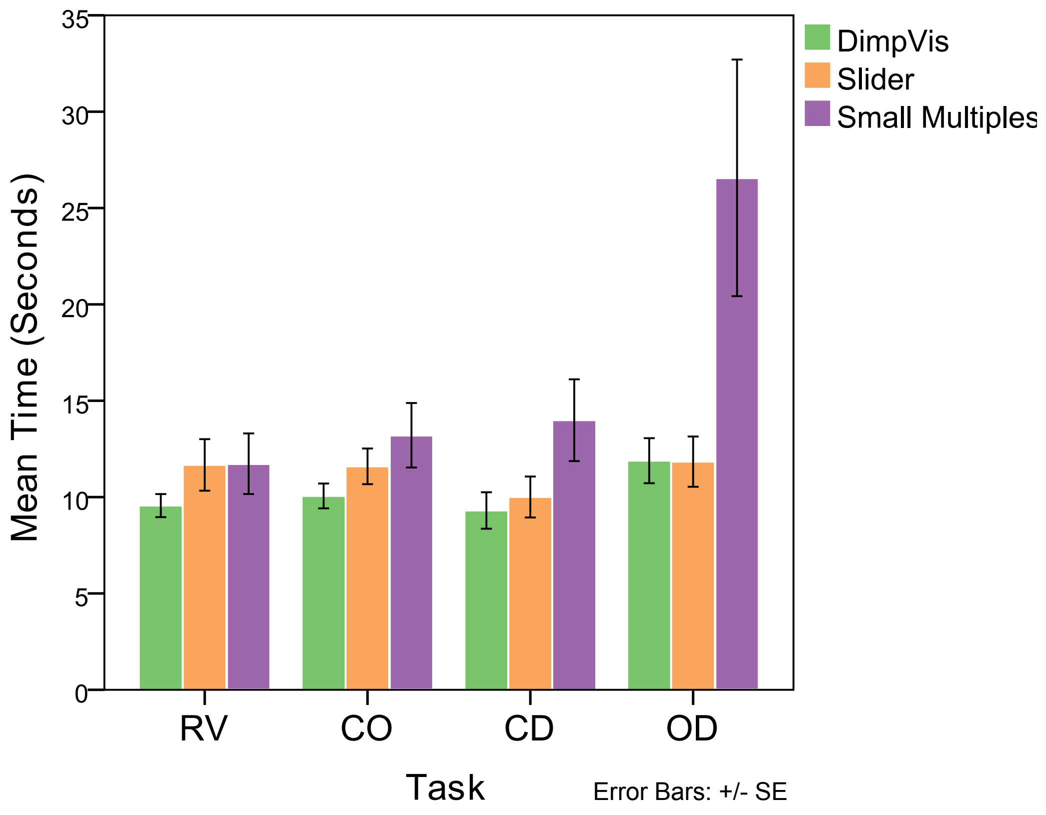

Fig. 4.5 Scatter plot task completion times by task type: retrieve value (RV), comparison (CO), characterize distribution (CD) and outlier detection (OD).

The results are summarized in Figure 4.5. The main effect of *technique* was significant ($p = 0.002$), with post hoc tests showing that DimpVis ($M = 10.2s$) and slider ($M = 11.3s$), were significantly faster than small multiples ($M = 16.4s$). However, the difference between DimpVis and slider was not significant. Therefore, H1 is only partially supported. There was also a significant main effect of *task type* ($p < 0.001$), with post hoc tests showing that RV ($M = 11s$), CD ($M = 11.1s$), and CO ($M = 11.6s$) were significantly faster than the OD task ($M = 16.7s$). However, the differences between RV, CO and CD were not significant, only partially supporting H2. Lastly, the interaction effect between *technique* and *task type* was significant ($p = 0.011$), due to the differences between DimpVis ($M = 11.9s$) and small multiples ($M = 26.6s$) in the OD task, as well as, slider ($M = 11.8s$) and small multiples in the OD task, as determined by post hoc tests. This result only partially supports H4, and H3 is rejected.

Small multiples required participants to scan through each image to locate the target point, resulting in slower completion times. Unsurprisingly, animation seemed to accelerate trend-based tasks, since times were faster for the CD and OD tasks using DimpVis and slider. The OD task was significantly slower using small multiples, and was tedious to answer, according to participants. This suggests that the small multiples technique becomes distinctly slower as the amount of moving points to observe increases.

Error Rate

Table 4.2 Scatter plot error rates produced by each technique, for each task type.

	RV	CO	CD	OD
DimpVis	0	1/36	2/36	0
Slider	1/36	1/36	3/36	4/36
Small Multiples	1/36	0	0	3/36

Overall, error rates were low for each technique (DimpVis= $3/144$, small multiples= $4/144$, and slider= $9/144$) and nearly uniformly distributed, therefore, no significant differences were found. Despite its relatively fast task completion time, the slider produced the highest error rate overall, especially seen in the CD and OD tasks. Thus, although horizontal dragging can be performed quickly, locating a specific moment in time when a change occurs is less precise. Unexpectedly, very few errors were produced by each technique in the RV and CO tasks, however for trend-based tasks (CD and OD), error rates were higher. Trend tasks were intended to be more difficult, as they require locating a specific time while interpreting the motion of a point, whereas RV tasks involve moving points directly to a known position.

4.2.2 Scatter Plot: Subjective Feedback

The majority of participants agreed that DimpVis was easy to use (see Figure 4.6). However, opinions on the loops were divided: half of the participants reported them as generally easy to use (some even found them fun), whereas the other half felt frustrated. One participant suggested that it would be nice to be able to fast-forward through ambiguous regions, because they slow down navigation. Some participants clarified that different techniques were more useful for different types of tasks. Specifically, two participants found that DimpVis was easier for completing the CD and OD tasks, compared to the slider. Lastly, participants expressed their excitement for DimpVis, that they “would use it to look at data,” it “increased engagement with the data," and that it could be “useful for teaching [charts to] students.”

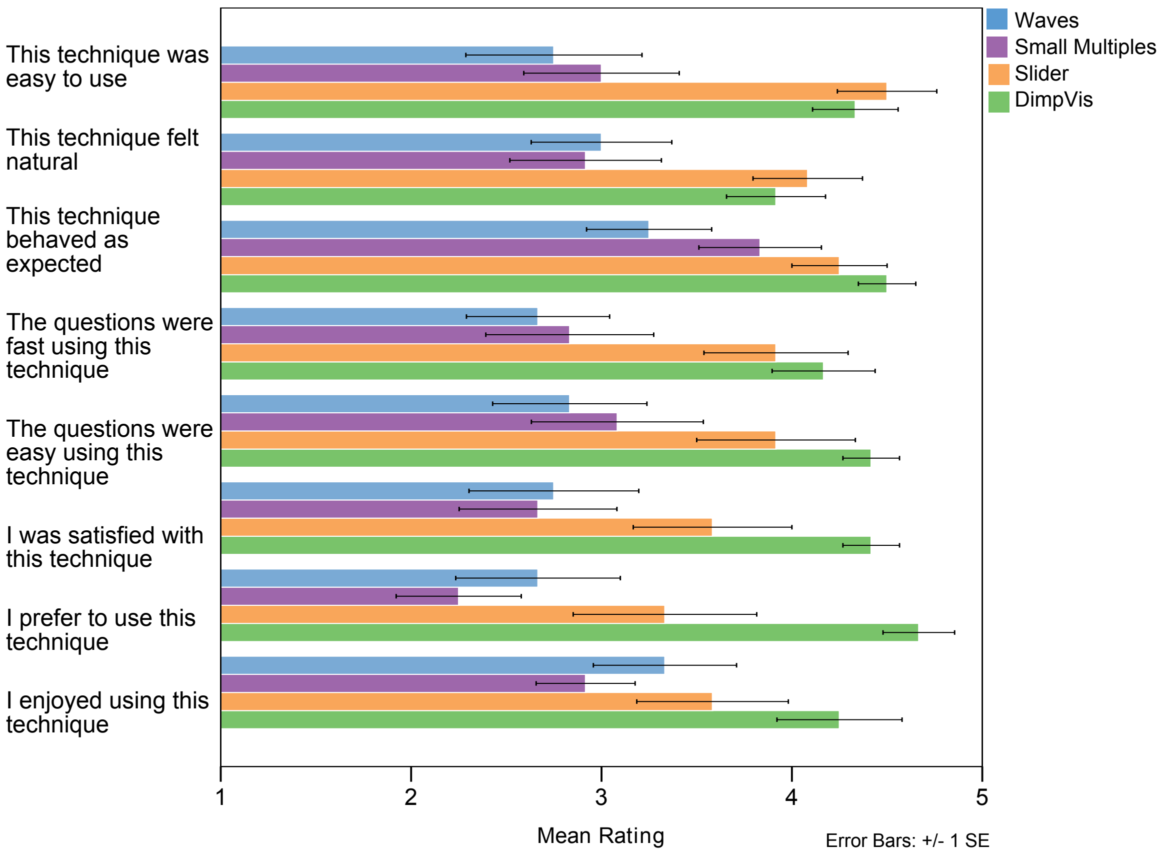

Fig. 4.6 Scatter plot subjective ratings.

Table 4.3 RM-ANOVA results for bar chart task completion times.

Factor	F-score	p-value
technique	$F_{2,22} = 3.064$	$p = 0.067$
task type	$F_{2,22} = 62.313$	$p < 0.001$
technique x task type	$F_{4,44} = 2.325$	$p = 0.071$

4.2.3 Bar Chart: Quantitative Results

Task Completion Time

A two-way repeated measures ANOVA with factors *technique* (3 levels) and *task type* (3 levels) was performed. The dependent variable, time (measured in seconds), was log-transformed to reduce skewing in the data caused by outliers. After transforming the data, Shapiro Wilk Tests indicated one variable was not normally distributed ($p < 0.05$), therefore one outlier was removed and replaced with the next value at least two standard deviations from the mean. Re-running Shapiro Wilk tests indicated that all variables were normally distributed ($p > 0.05$). Mauchly's test showed that the assumption of sphericity was satisfied for all main effects. We adjusted significance values for all post-hoc pairwise comparisons using the Bonferroni correction.

The results are summarized in Figure 4.7. On average, slider was the fastest ($M = 13.2s$), closely followed by DimpVis ($M = 13.5s$) and then small multiples ($M = 15.7s$). However no signficant differences were found for *technique*. There was a significant main effect of *task type* ($p < 0.001$), with post hoc tests showing that the differences between the CO ($M = 17.2s$), CD ($M = 14.6s$) and RV ($M = 10.6s$), tasks were all significant. No significant interaction effect between *technique* and *task type* was found. Therefore, H4 is only partially supported and all other hypotheses are rejected.

The CO tasks were consistently the slowest across all interaction techniques. During these tasks, we noticed that some participants stepped away from the display to compare the bars when using slider and DimpVis, which may have lead to slower times. This may also indicate that the screen size selected was too wide.

Error Rate

Overall, error rates were low for each technique (DimpVis= 0, slider= $4/108$ and small multiples= $7/108$), and nearly uniformly distributed. Therefore, no significant differences were found. Error rates varied between each type of task (RV= $1/108$, CO= $4/108$ and CD= $6/108$) and DimpVis produced no errors on any tasks.

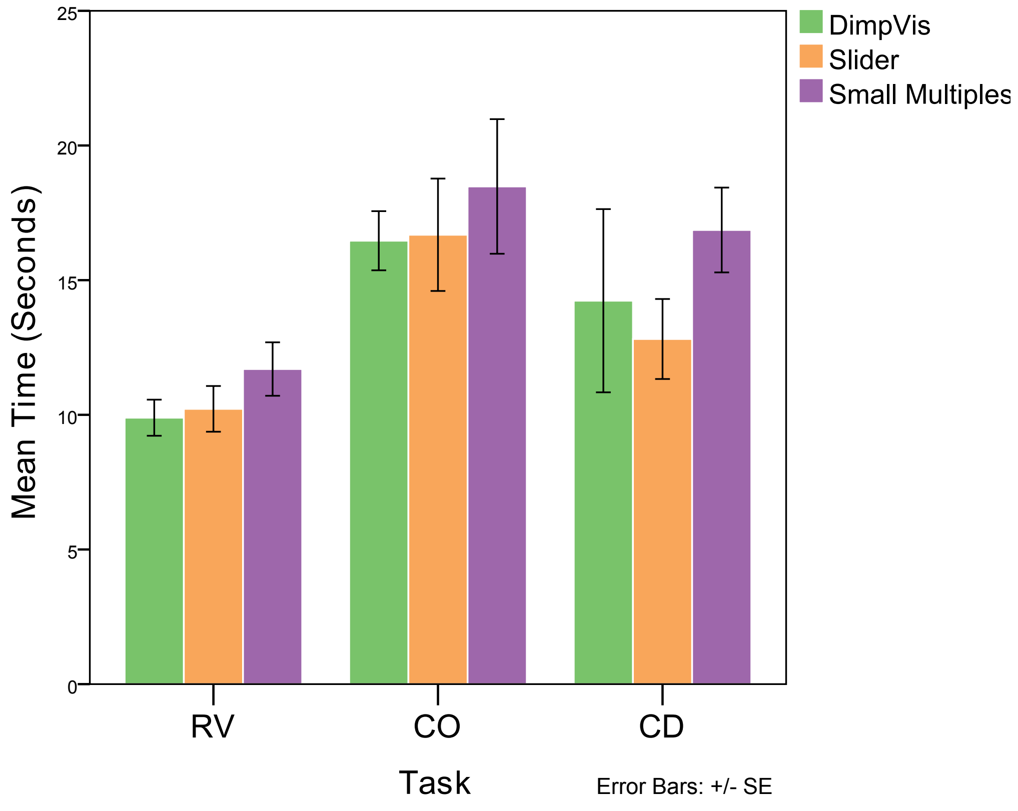

Fig. 4.7 Bar chart task completion times.

Table 4.4 Bar chart error rates produced by each technique, for each task type.

	RV	CO	CD
DimpVis	0	0	0
Slider	0	0	4/24
Small Multiples	1/24	4/24	2/24

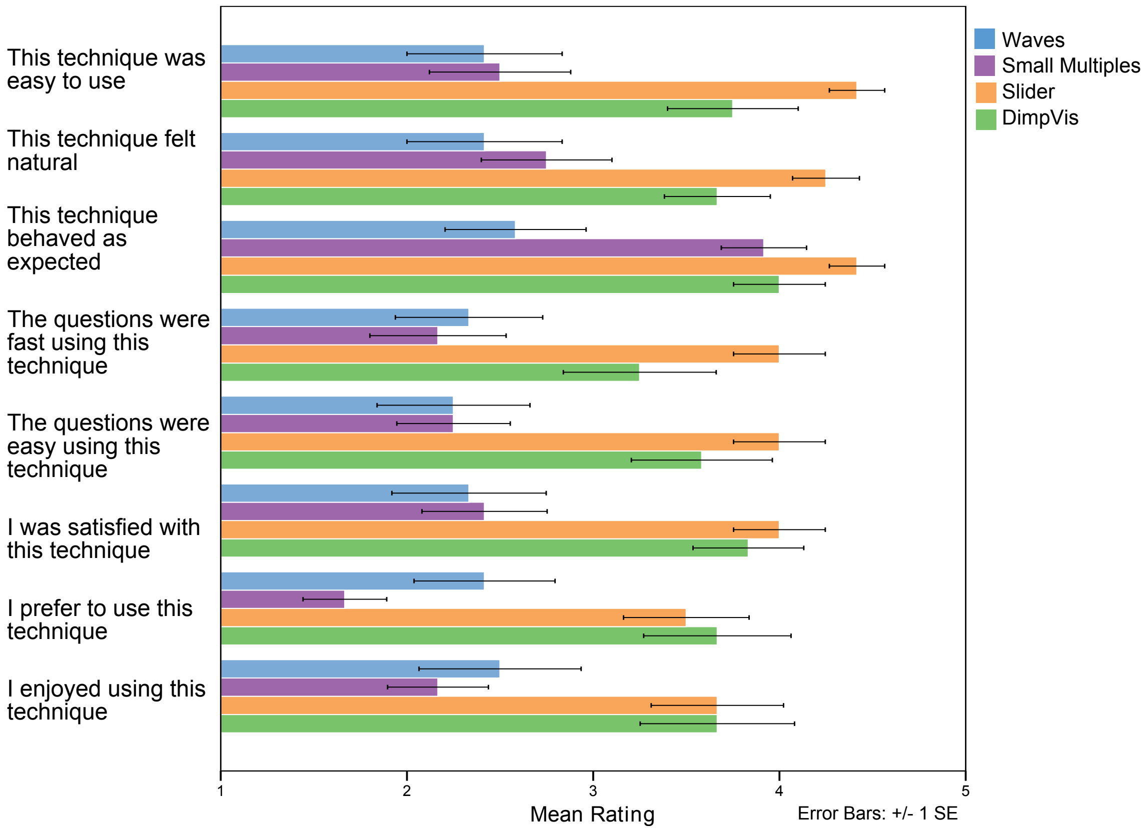

Fig. 4.8 Bar chart subjective ratings.

4.2.4 Bar Chart: Subjective Feedback

The subjective feedback indicates participants generally preferred DimpVis over the small multiples (see Figure 4.8). Three participants mentioned that dragging the bars was confusing. One participant stated that it was “difficult to understand the simultaneous height and time change.” The majority of the feedback pertained to the wave interaction detour. Only three participants commented that the waves were easy to use. The remaining seven participants expressed various concerns regarding the usability of the waves, such as feeling “lost in the wave,” or how the wave “refused to respond during dragging.” Overall, majority of participants stated that they preferred the slider, mainly because it was easier to use and learn quickly (e.g., horizontal dragging motion).

Despite being informed that dragging must follow the path, we observed at least four participants attempting to drag away from it, multiple times during the tasks. For instance, in the RV task, they would try to drag directly to the height, even though the path indicated a different dragging direction. Five participants also used a second finger to either drag other bars not involved in the task or mark a significant spot on the chart, such as another bar in CO task, or the target height of an RV task. Additionally, some interesting attempts to accelerate navigation were observed, such as: swiping up to reach far heights, horizontal dragging along waves or shorter bars, and even tracing the wave. These may suggest ways to refine the design.

4.2.5 Exploratory Period

For the scatter plot, a dataset representing total internet users for some of the world's major economies from http://gapminder.org was used. For the bar chart, we used a dataset showing total enrollment in different programs at our university from http://cudo.cou.on.ca. These datasets were chosen as they may be interesting to our participant population.

Common strategies were observed for exploring the data. Some participants (3-bar chart and 4-scatter plot) would focus mostly on reading the trends of individual points or bars using the hint paths, performing less temporal navigation. However, other participants (3-bar chart and 5-scatter plot) preferred to drag only a few points/bars, while examining the motion of other data objects in the visualization. Lastly, the remaining participants almost evenly divided their attention between the hint path and dragging; by first examining a hint path and then dragging along it to explore time (6-bar chart and 3-scatter plot). Notably, one participant used a storytelling approach, where they narrated the trends of the points and bars, while dragging them. The hint paths and interaction technique seemed to supplement the story.

Subjective feedback on the hint path was uniformly positive across both visualization types. The hint path was rated as beneficial ($M_{scatterplot} = 4.8$, $M_{barchart} = 4.7$), helpful ($M_{scatterplot} = 4.6$, $M_{barchart} = 4.6$) and useful ($M_{scatterplot} = 4.8$, $M_{barchart} = 4.2$) during exploration. Generally, participants did not find the hint paths distracting ($M_{scatterplot} = 1.3$, $M_{barchart} = 1.4$) or confusing ($M_{scatterplot} = 1.4$, $M_{barchart} = 1.3$) when exploring the visualizations.

All participants agreed that DimpVis was a suitable interaction technique for touch-screens, mainly because it seemed to "enhance engagement with the data." Three participants suggested that a mouse may be more suitable for navigating the loops and waves, because they felt more precision was needed for navigating along them. The visual complexity of the detours may give the impression that more precision is required (e.g., following the sine waves closely), when in fact they require a similar dragging motion as the regular hint paths.

4.3 Design Implications

The main goal of our evaluation was to determine the benefits of using DimpVis for answering questions targeting the visual space, and compare its performance to the time slider and small multiples technique. DimpVis for the scatter plot was significantly faster than the small multiples and subjectively preferred by participants. DimpVis did not significantly out-perform the time slider, however there was no loss in time or accuracy. This null result suggests that DimpVis may be useful for supporting some object-centric tasks that are cumbersome using the slider or small multiples, without any loss in performance.

Based mainly on findings from the evaluation, we note some implications for the design of DimpVis for the scatter plot and bar chart:

Hint Paths Aligned with Dragging Direction: We expected that DimpVis for the bar chart would to be easier to learn and use, because it requires only vertical dragging and a single changing visual variable over time, whereas the scatter plot has two changing data dimensions over time. All participants appeared to use DimpVis for the scatter plot consistently: dragging along the path using one finger. Whereas, for the bar chart, a diverse set of unsuccessful interaction actions were observed, including using a second finger and dragging in a direction opposite to the path. This indicates that participants may have expected DimpVis for the bar chart to have different capabilities, suggesting that dragging the bars was a less intuitive interaction than dragging the points. The mix of vertical dragging and horizontal hint path translation seems to cause some confusion, as opposed to the scatter plot's stationary hint path.

This might suggest that restricting dragging to the direction of the hint path is preferrable. This is a difficult design challenge, as it competes with our design goal (D3) of keeping the finger on the bar (meaning only vertical motion is possible for bar charts). Given these constraints, for bar charts this would mean hint paths should clearly indicate vertical motion. One approach is to provide vertical arrows, scaled to the amount of change, which indicate at any instant in which temporal direction dragging will move time (however, this would not support fast-forwarding). Alternatively, our flashlight hint path may be a solution as it allows for direct dragging to any year of interest . However, navigation is no longer restricted to following the temporal order.

Multi-touch Ambiguity Resolution: The usability of the wave interaction detours was a concern raised by some participants. This may have been due to the lack of an anchor to explicitly indicate the position along a wave, which was provided for the loop. Also, better tuning of the tolerance region around the peak of the wave may help prevent unwanted direction reversals. However, a more complete solution may be to take a different approach to temporal ambiguities, such as relaxing our goal of interaction consistency (D4) in favour of introducing a second finger for scrolling through time steps where the value does not change.

Provide Time Line and Flashlight Hint Paths: Initially, we selected the time line hint path design because clearly illustrating temporal trends was considered an important requirement for guiding temporal navigation (D1). However, during our evaluation, some participants attempted to drag bars directly to a desired height, an action that may be better supported by our flashlight hint path design (Section 3.1.1). Different types of tasks may be better supported by different hint path designs. Additionally, both detailed (dragging) and accelerated (fast-forwarding) temporal navigation are important (D2), since one participant found the interaction detours slowed down navigation when completing the tasks, and wanted to quickly skip through them. The fast-forwarding feature was not provided in the evaluation, as we wanted to focus on the dragging interaction.

We initially hypothesized that the time line hint path could sufficiently support direct queries such as: “Was this bar ever at 500?”. However observations of participants using DimpVis for the bar chart suggested that the type of navigation may be task-dependent. Therefore, we designed flashlight hint paths for the scatter plot and bar chart (Chapter 3).

Chapter 5

Glidgets

Glidgets was originally intended as an application of DimpVis to dynamic graph visualizations. Since dynamic graphs are different than the types of visualizations explored during the design of our DimpVis examples (e.g., multiple, changing visual variables), we distinguished Glidgets as its own technique. However, to relate these two techniques, Glidgets demonstrates another example of DimpVis, for non-spatial visual variables.

Glidgets (glyph widgets) consists of subtle, yet informative glyph representations of low-level node and edge changes and corresponding interaction techniques for issuing temporal queries directly on dynamic graphs. Specifically, the changes visualized by our technique are: node/edge presence (disappearance and reappearance) and node degree (number of incident edges). The topological changes are visualized in interactive time line glyphs, acting as time sliders embedded in an edge or node, used to invoke object-centric navigation. This allows a user to first query graph element changes and then investigate those changes by directly controlling temporal navigation, while remaining focused on the element of interest.

In this chapter, we discuss the design of Glidgets and present a use case scenario illustrating how Glidgets can be used to explore a dynamic social network.

5.1 Design

Glidgets consists of a set of interactive glyphs for visualizing changes of graph elements and guiding object-centric temporal navigation. The Glidgets design has three main components:

1. Directly query an element or a group of elements of interest to ask hypothetical

questions about changing elements such as “Are these two nodes ever connected over time?” (Section 5.1.1)

2. In response to a query, a visualization of element changes is presented in a detailed glyph (Section 5.1.2)

3. Use the glyph as a guide for object-centric temporal navigation, to explore the changes visualized by the glyph (Section 5.1.3)

Below, each listed component is discussed in detail.

5.1.1 Directly Querying Element Changes

The user can select any element, or multiple elements, and receive immediate visual feedback, showing how the element changes over time. To select elements we use gesture-based sketching interaction with a stylus. Mouse input lacks the necessary control for sketching and touch input lacks precision for selecting small objects, as the hand can occlude the visualization [30]. Precision is particularly important when selecting elements from graph visualizations, where visual elements tend to be small and densely packed. In addition, direct interaction with elements using touch or a stylus can bring a feeling of direct connection or embodiment to the analysis process [9, 21].

Our technique is designed to work alongside existing techniques that can effectively portray high level changes (e.g., animation) or visualize the union of all changes between time slices (e.g., difference highlights). Consequently, we designed a set of querying capabilities that can answer questions targeting the changes of individual or sets of graph elements in the context of the whole graph.

The core querying capabilities were designed to answer following types of questions:

Q1 Element(s) existence Whether or not an edge or a group of elements ever exists in the dataset. **Example questions:** *Are these nodes ever connected? Does this path of nodes ever exist?*

Q2 Changes of single elements Observing how a single element changes over time. **Example questions [4]:** *How does a node’s degree change over time? When does a node disappear and re-appear?*

Q3 Changes of element groups Observing how groups of elements change over time. **Example questions:** *When does this group of nodes appear together over time? When does this group of edges appear together over time?*

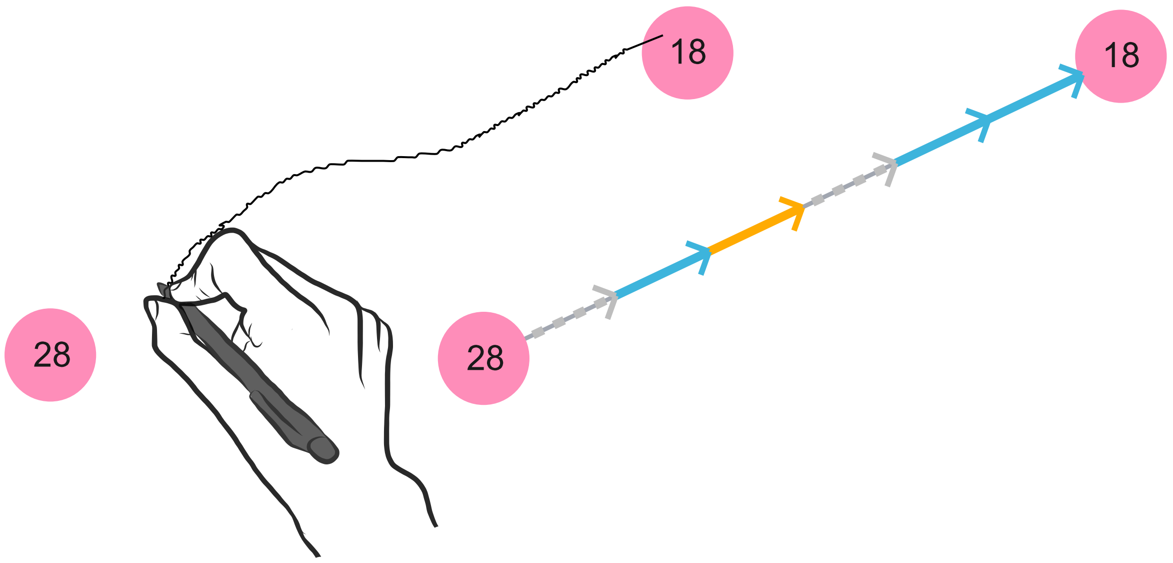

Fig. 5.1 An edge glyph is revealed by sketching a line between two nodes.

To answer these questions, we created two different types of queries: single-element (Q1 and Q2) and aggregated (Q1 and Q3). Single-element queries are issued by simply selecting the element of interest to reveal its change glyph (Section 5.1.2). For nodes, selection is done by tapping it. To select an edge, a line can be sketched between any two nodes (Figure 5.1(left)). Note that the line need not be straight — it is simply a stroke starting at one node and ending at another, allowing for flexible query sketching. Using the same selection techniques, multiple elements can be selected, and are correspondingly added to or removed from an aggregated query. After an element is selected, its change glyph appears. Aggregated change glyphs are updated according to the query elements.

5.1.2 Visualizing Temporal Changes of Elements

Due to the temporal nature of dynamic graph data, it is common for elements to be added or removed from the network, at different moments in time. For instance, in a social network, edge addition and removal shows the formation and breakage of friendships among people in the network. Node degree is an important social network analysis metric to measure the level of engagement of a person in the network. We visualize these three low-level changes in change glyphs: node and edge presence (addition and removal) and node degree.

Highlighting techniques can draw attention to and help users understand changing graph elements (e.g., [5, 6]). Change glyphs are designed to represent a time line of changes, highlighting the presence of graph elements.

Our two main design goals for the glyphs were:

D1 Minimal, yet useful representation Graph visualizations tend to be cluttered, due to the presence of nodes and multiple edges. Therefore, it is important for any additional visual elements to not clutter the graph. However, graph elements tend to be small, so the glyphs should be legible among the surrounding elements.

D2 Intuitive, time-line visual metaphor The structure of the glyph should convey a time line of changes, visually separating each time slice. The time line arrangement of the glyphs should integrate well with the graph element. The temporal direction should also be clear, since the glyph guides temporal navigation.

Presence only has two values to encode: present or absent. We visually encode presence with colour, grey for absence and blue for presence. We explored alternative designs for encoding presence, such as altering the geometry of the glyph (e.g., as a sine wave). However, different hues can be easily distinguished (D1) and do not depend on the orientation of the glyph (D2). The glyphs are structured as small time lines evenly divided into segments for each time slice. Marks are added along the segments of the glyph to explicitly indicate the time slices (D2). To integrate the glyphs with the graph elements, they are placed around or on top of existing graph elements (D2). In the next sections we discuss the node and edge glyph designs and another version of the glyphs for conveying an aggregation of changes in presence across multiple elements.

Node Change Glyphs

A clock metaphor is used to arrange the time slices. The time line begins at the top of node and is wrapped around the node (D2, Figure 5.2). The glyph is divided into equal-sized annulus segments, coloured according to the node's presence. One segment is assigned for each time point, and the start points of time intervals are marked by a darker, coloured line at the start of glyph segments (D2). Our design is similar to the pie chart glyphs used in the DGD system, however they do not show how a node metric varies over time [42]. Alternatively, to show presence and degree, small visualizations of changes could be embedded inside the node, such as a line chart (e.g., [51]) or a pie chart (e.g., [1]). However this can interfere with existing visual encodings applied to nodes (e.g., node colour showing group membership, node labels) mak-

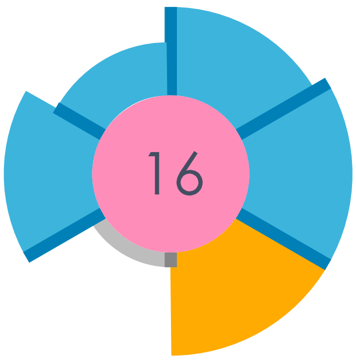

Fig. 5.2 A node's glyph is revealed by tapping it. This glyph shows that node 16 disappears once, at time 3. The orange segment indicates the current time slice.

ing it less visualization-independent.

The varying heights of the glyph segments encode the relative node degree. The height is proportional to the ratio of current degree at a segment and max degree of the node over time. The maximum and minimum heights are the same for all nodes. Varying the height forms a circular barchart, similar to Nightingale's rose (coxcomb) diagram [39], making differences in degree distinguishable even though the glyph is small (D1). We also considered interpolating the height differences to smooth the degree changes, forming a spider plot. However this implies that node degree is continuously changing, whereas the coxcomb-like design is more truthful to the data, by presenting node degree values only at specified time points.

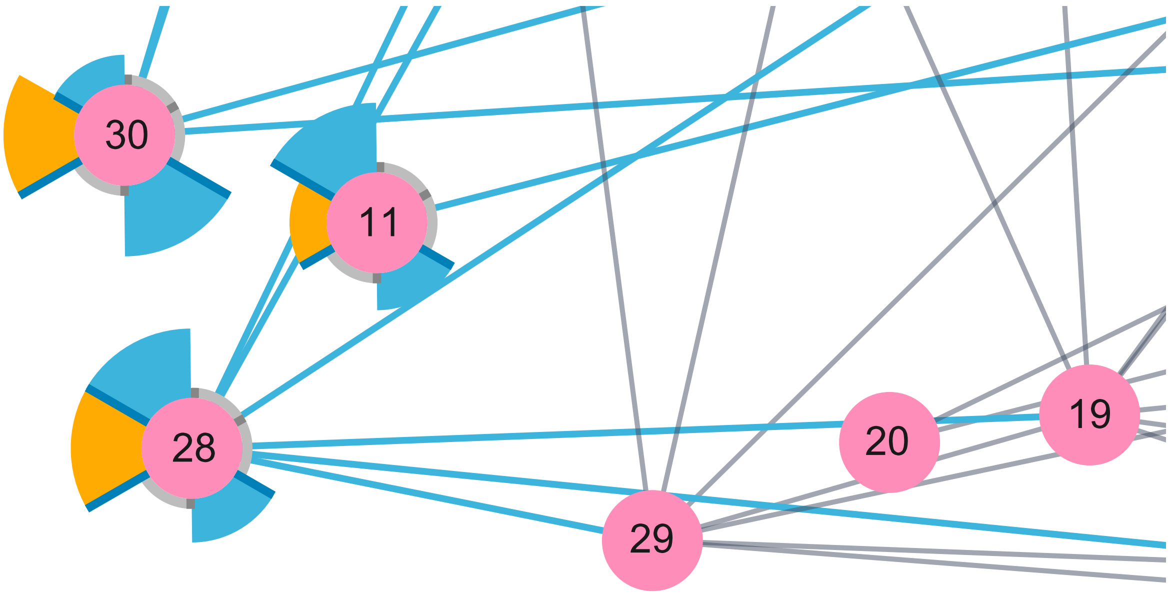

Fig. 5.3 Aggregated node change glyphs are shown for three selected nodes. Blue segments indicate all nodes are present at that time slice, while grey segments indicate at least one node in the group has disappeared.

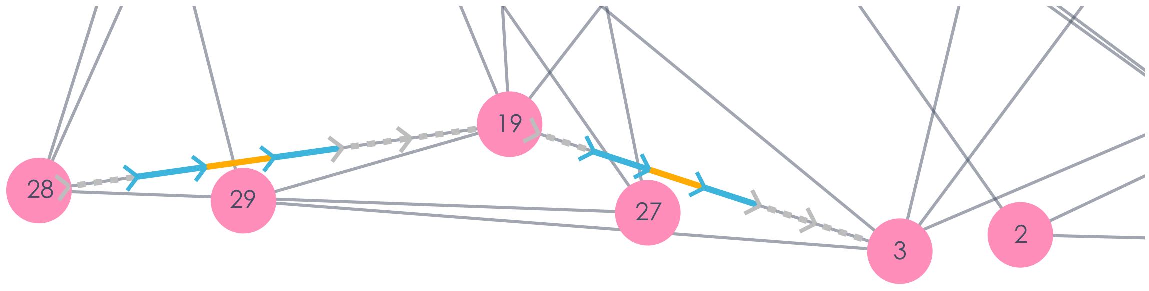

Fig. 5.4 Aggregated edge change glyphs are shown for two selected edges. Blue segments indicate all edges are present at that time slice, while the grey segment indicates one of the edges has disappeared.

Edge Change Glyphs

A linear time line metaphor is used to arrange the time slices (i.e., time begins at left-most node) and the glyph is drawn as a temporary highlight on top of the edge (D2, Figure 5.1(right)). The segments of the glyph are equally divided, one per time step. The beginning of each segment is marked with an arrow pointing forward in time, indicating the temporal direction (D2). For any two nodes, the arrows always point toward the right-most node, matching the layout of the horizontal time slider. If the edge is vertical, then the arrows point towards the top node. The edge change glyphs are coloured according to the edge's presence using the same blue and grey colouring as the node glyphs. When the edge is not present during a time step, a grey dotted line is drawn. The dotted line in the glyph allows an analyst to see the underlying edge disappear during temporal navigation (Section 5.1.3) by tracing the edge glyph. We also considered drawing a coloured highlight around the edge. However this glyph would be thicker, occupying more screen space (D1).

Aggregated Change Glyphs

When mutliple graph elements are selected, an aggregated change glyph is created (Figure 5.3 and Figure 5.4). The visual encoding of this glyph is identical to the single-element change glyphs, however it is recoded to show when multiple elements appear together over time, or, the *intersection of presence*. This is analagous to an AND query about presence at each time step. Therefore, a blue segment means all selected elements are present at that time slice, whereas a grey segment means at least one element has disappeared. We designed this glyph to support exploring and comparing temporal changes of element groups.

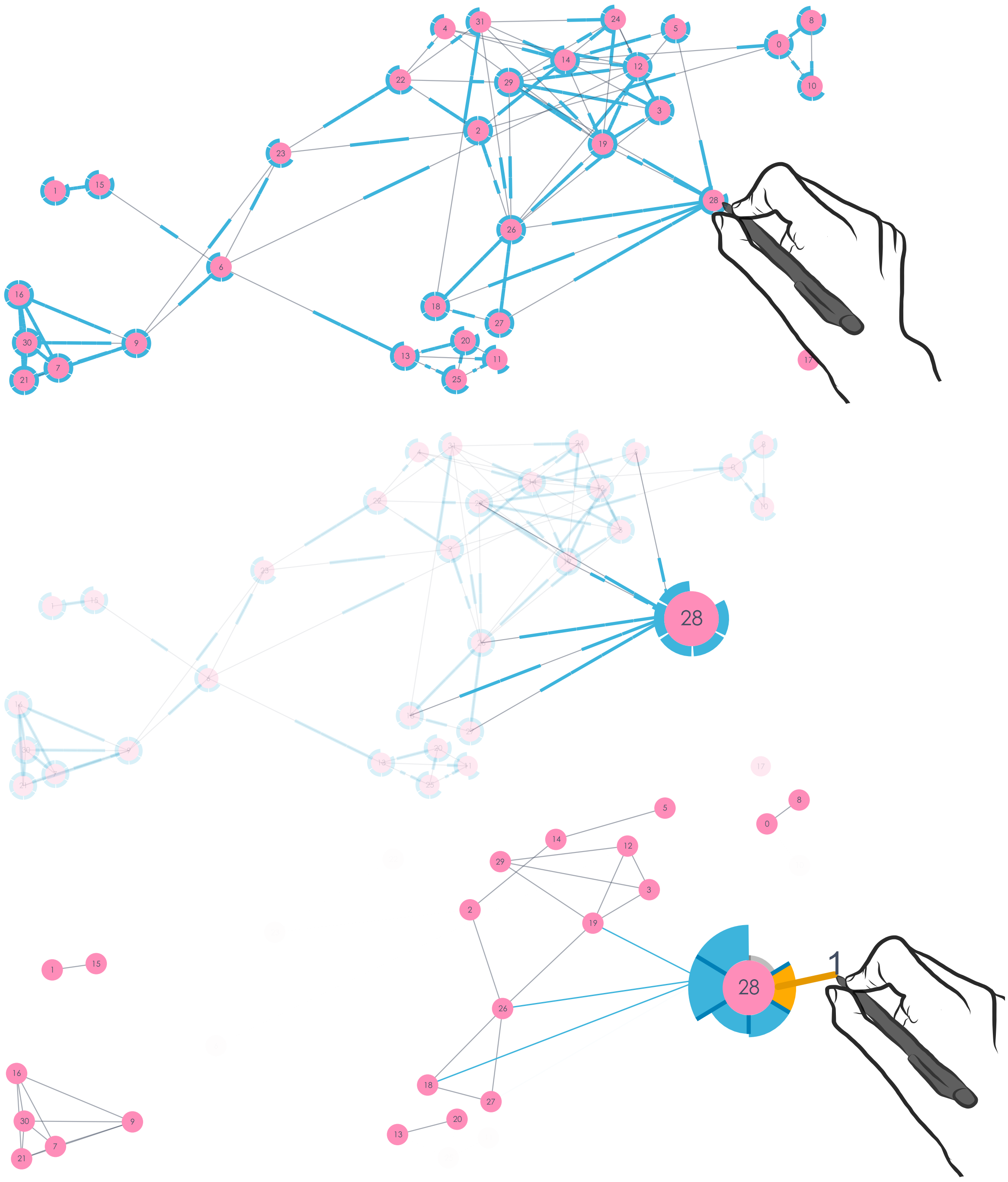

Fig. 5.5 The global view shows the global glyphs of all elements (top). Elements can be interactively selected (middle), and the user can transition directly to any time point by dragging a glyph, which also cancels the global view (bottom).

Global Presence Change Glyphs

In order to select an element and reveal its glyph, the element must be present in the view. Moreover, the independent change glyphs of elements cannot be visible at once because glyphs are automatically aggregated as mutliple elements are selected. To support global analysis of changes over time, we created a *global view*. This view uses *global change glyphs* to show presence changes of all elements, across all time steps (Figure 5.5 (top)). The global change glyph design is similar to the normal change glyphs, except only presence of an element is represented as blue segments, and the grey regions (absence) are left blank. This makes the glyphs smaller and conserves some space. Global presence glyphs can be used to locate elements of interest by presenting a comprehensive overview of presence changes among elements.

Elements can be selected in the global view, and the change glyphs of selected elements are highlighted by fading all other elements. When multiple elements are selected in the global view, the change glyphs remain independent. That is, they are not aggregated. Originally, our global presence glyphs were revealed as a non-interactive, temporary overlay that was disconnected from the regular graph view. However after pilot testing, participants found the global presence glyphs useful, mainly as a starting point for locating elements. They wanted it to be interactive, particularly to jump directly to a time point when an element is present. Therefore, we decided to add interaction techniques to the global view enabling easy transitioning between the global and regular graph views, and interactive selection of elements (Figure 5.5 (middle)). Thus, the global view can be used to visually locate an item of interest and jump directly to any time step. For example, a user can select a node in the global view and the other elements are faded to simplfy the view, bringing attention to the node's global glyph. Then, transitioning out of global view can be done by activating an embedded slider on the node (Figure 5.5 (bottom)), as described in Section 5.1.3.

5.1.3 Object-Centric Navigation

We designed object-centric navigation for exploring dynamic graphs by embedding time sliders in the change glyphs which can be invoked on any selected element. This way, navigation is performed where the visual analysis takes place. For example, if a user is examining a node's degree changes on its glyph and wishes to see the details of connection changes over time, they can use the embedded time slider to scan through time while focusing on the node.

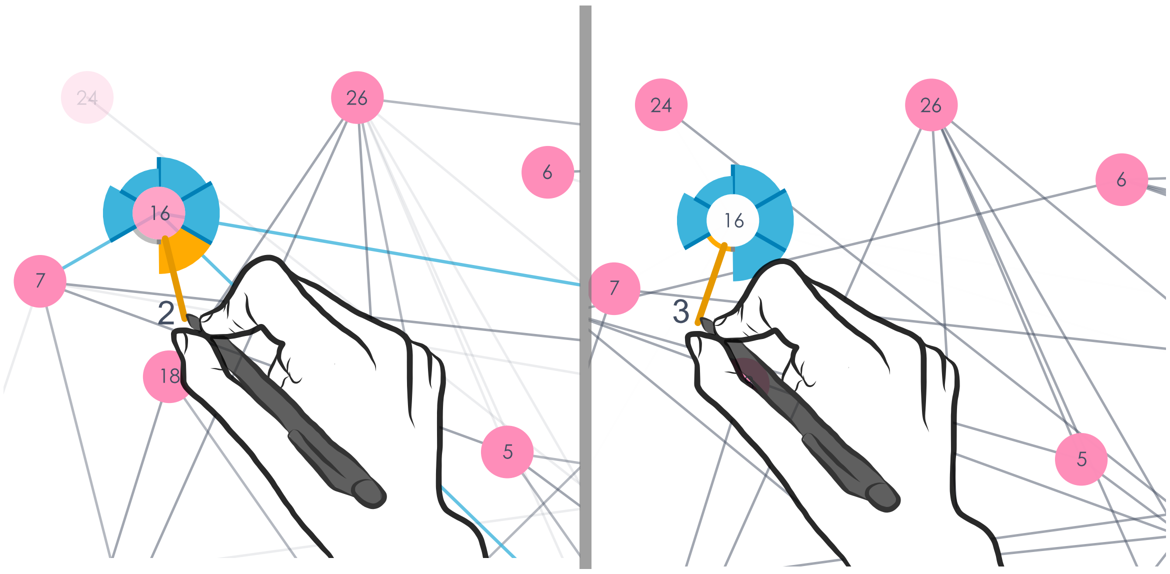

Fig. 5.6 Dragging along the glyph moves in time and other elements are faded in and out (left). When a selected node disappears, its glyph and label remains visible until it is deselected (right).

Embedded Time Sliders

When an element is selected, the user can drag along the change glyph to navigate time. Thus, the glyph becomes an interactive, embedded time slider. Dragging allows for intuitive, user-controlled navigation, where the user's actions correspond to the immediate visual outcome [53] and the rate of animation is synchronized to the rate of dragging, creating interaction compatibility [9]. Furthermore, pen input provides a high level of control, when following small objects along arbitrary paths [30]. If an element disappears during dragging, the glyph remains visible providing continuous temporal navigation through periods where the selected element is not present. The glyphs show relative changes, therefore, the embedded time sliders can be used to explore or verify the details of the changes, while remaining focused on an element. In the case of an aggregated query, the glyphs show the intersection of presence. Using the embedded time slider, a user can review the detailed presence information for the selected elements. For example, grey regions indicate that at least one element has disappeared. The embedded time sliders can be used to identify which element(s) disappeared.

Node Time Slider Since the node's glyph is arranged using a clock metaphor, when dragging around it, the temporal direction is designed to be intuitive: rotate clockwise to go forward in time (Figure 5.6(left)). This is intended to be similar to moving the hands around a clock. To improve visual tracking during navigation, the node's

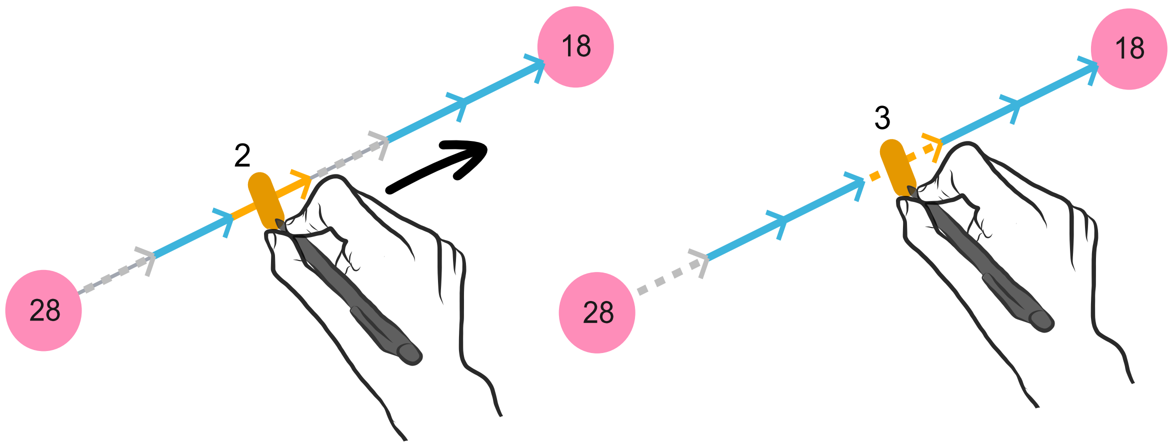

Fig. 5.7 Dragging along an edge glyph moves in time. It is shown here that the edge between 18 and 28 disappears at time 3.

incident edges are highlighted in blue. If a node disappears, its label remains visible, in order to identify it (Figure 5.6(right)).

Edge Time Slider The temporal direction when dragging along an edge glyph is always left-to-right, to move forward in time. However, as edges are sometimes steeply vertical, the time direction is also indicated by the arrows (Figure 5.7).

Navigation Cues

When dragging along an edge glyph, a short sliding bar appears perpendicular to the glyph, and can be dragged along the glyph, similar to a traditional time slider. When dragging around a node, an elastic tether is drawn, connecting the pen tip to the glyph. As the user is navigating in time, the entire glyph segment is highlighted in orange to mark the current time position and visually reinforce that the data is constructed from discrete time steps (Figure 5.8). During pilot testing, participants suggested a label for the time slice be added, so that they did not have to look at the horizontal time slider to know the current time step. Thus, the current time slice label is drawn near the pen tip as an explicit time indicator during navigation.

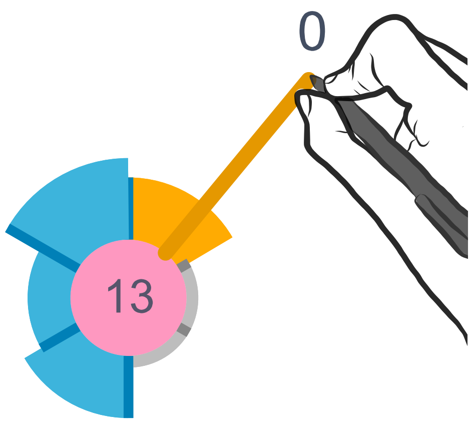

Fig. 5.8 When navigating time using an embedded slider, the current time slice is displayed next to the pen tip and the corresponding segment on the glyph is highlighted in orange.

The time sliders may be acquired at any point in time, jumping time to that position. For example, tapping an edge slider in the middle will initiate time navigation at the middle time point in the data. This fast navigation technique was designed to allow immediate movement to time points with interesting presence information shown on the glyphs.

Regular Time Slider

Since our technique focuses on observing local, element-level changes, it is important to include a regular time slider, in order to observe high-level changes occurring in the graph. The time slider is synchronized with all embedded sliders, to show the current location in time.

5.1.4 Graph Layout and Representation

We used an unweighted, undirected node-link representation, where nodes (circles) represent the individual entities in a network, and edges (lines) are drawn to connect nodes, showing a relationship between them. In our first design iteration, the dynamic graph was created using a force-directed layout generated independently for each time slice and node position was animated during temporal navigation. After early testing we found the node movement to be distracting and inconvenient when interacting with the embedded time sliders. In order to overcome the interaction challenge, we temporarily fixed the node positions during dragging and then animated them to their new position when the dragging ended. However, it was easy to lose track of the node in focus even when the glyph was still visible. After releasing the slider, it was disruptive to move the pen to the new node position in order to continue using the slider.

In our current prototype, we calculated a single layout, using a force-directed algorithm on the union graph of all time slices (all nodes and edges ever present). This is a form of the "flipbook" technique, used to display dynamic graphs [36]. After calculating this layout, the nodes are fixed in position, unless manually moved by the user.

Cubic ease in and out functions are applied to nodes or edges that are fading in and out. The animation timing function is centered on the beginning of time step, with the complete animation taking place over one third of the time step. Since temporal navigation is always performed by dragging a time slider, the fading speed is directly controlled by the user. We also explored other animation timings and found that cubic functions focused the transition closely to the boundary between time slices, whereas, linear was too gradual to attract visual attention to fading elements.

Glidgets was implemented in Java, rendered with Processing [43]. The graph layout was generated using the force-directed layout (Fruchterman-Goldman algorithm) with all default parameters, provided by the JUNG library [29]. The interactivity was designed for pen input, using a Lenovo x220 tablet laptop, running Windows. Glidgets has also been tested on our 3m Anoto pen-enabled wall display and is compatible with any Windows Touch-compliant systems.

5.1.5 Public and Lab Demonstrations

Glidgets was demonstrated at the 2014 GRAND conference, where useful feedback and suggestions were received from researchers. Many commented on the novelty of the technique, stating that is was an interesting temporal navigation method. However, some were skeptical about the scalability of the glyphs, when the time line increases. We noticed that Glidgets required a learning curve, as some people spent quite some time figuring out how to use each feature.

We demonstrated the earliest design of Glidgets to some colleagues, where we obtained feedback regarding the visual design. For example, aggregated glyphs were originally applied by holding down a key, which was found to be a complex and made the interaction feel less fluid. This was changed to automatically applying aggregation during multiple element selection, such that all interactions could be performed with the pen. The original design of the global glyphs portrayed the proportion of presence and absence across all time points. For example, the node global glyph was a pie chart surrounding the node with two segments showing presence and absence. The global glyphs were not found to be useful, since the presence proportion could be approximated using the regular glyphs, and the global glyphs were generally difficult to read and compare. Without increasing the amount of space for drawing the glyph, we changed the global glyphs to represent detailed presence information.

5.2 Use Case Scenario

Glidgets can be used to investigate temporal changes of graph elements. The change glyphs allow for fast and easy analysis of how an element changes over time, and the embedded sliders support direct navigation to a time point of interest on the glyph. Suppose Spencer is analyzing the Van De Bunt social network, where the dataset represents friendships among undergraduate students over time [56]. In particular, he wants to discover and

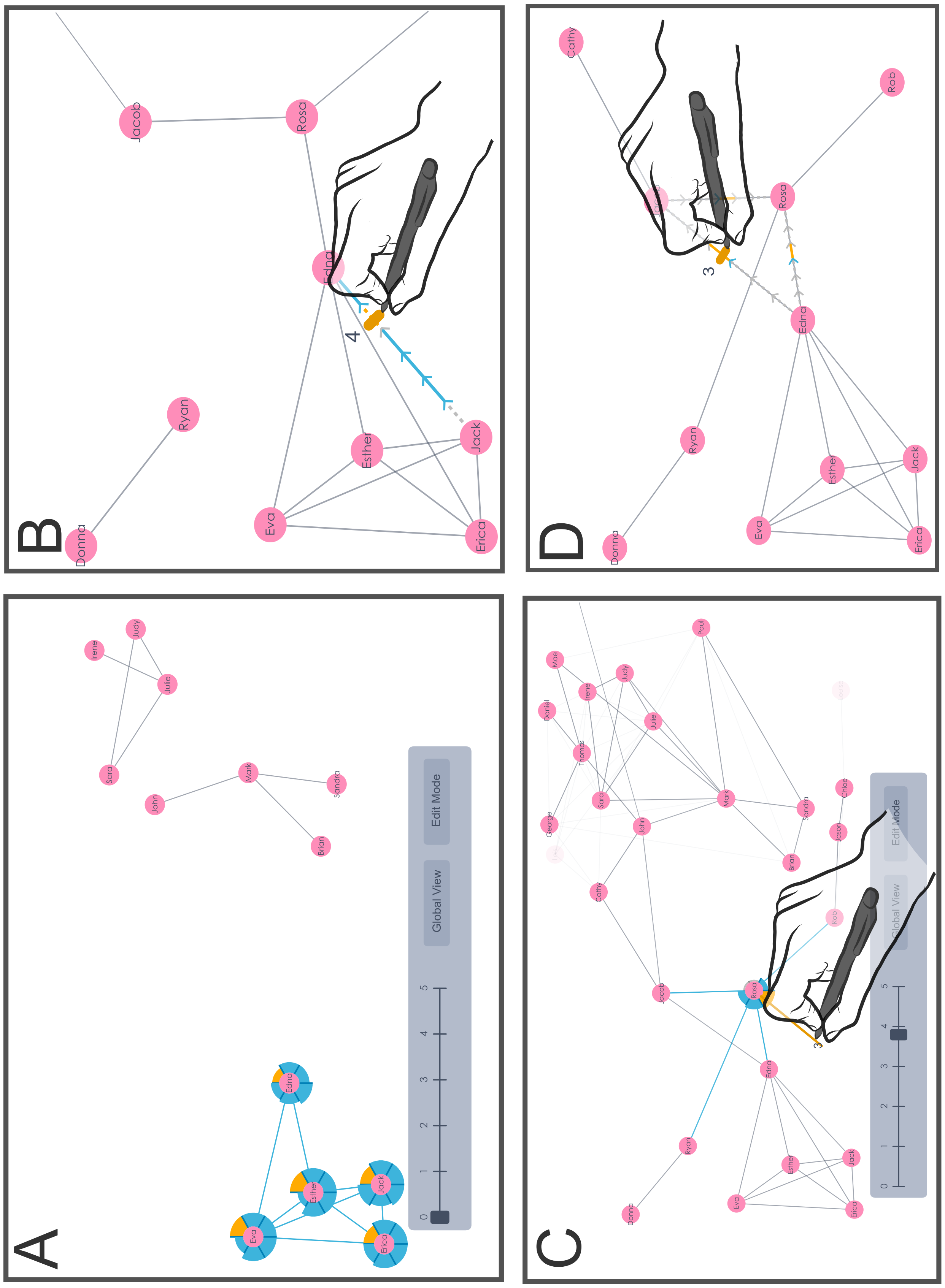

Fig. 5.9 Screenshots illustrating parts of the scenario: A) The original group of nodes and their glyphs, B) Investigating the friendship between Jack and Edna, and the emergence of Rosa, C) Dragging along the node glyph of Rosa to find her new friends, and D) Aggregating edge glyphs to find a small clique at time 4.

investigate groups of friends within the network, including when friendships in groups form and end, as well as people who are added and removed from the groups.

The Van De Bunt network consists of 31 nodes, 111 edges, and 6 time slices. For illustrative purposes, the nodes were artificially labeled with human names.

Spencer begins his analysis by examining the global view to search for nodes of interest. He scans the view to find people that are always present, indicated by a solid blue highlight and selects those nodes as he finds them. After selecting a group of nodes, he transitions directly to the regular graph view at time 0 using an embedded node time slider.

He notices that the group of people actually forms three, separate sub-groups of friends. He decides to focus on one of these sub-groups, particularly the one containing people with higher node degrees (more friendships), as indicated by the node glyphs. Therefore, he hides the node glyphs of the other two groups, by de-selecting them. The group of interest consists of five people: Eva, Esther, Jack, Edna and Erica (Figure 5.9(A)). He notices that not all members of the group are friends initially, at time 0. Specifically, Edna is not friends with Erica nor Jack.

He wants to investigate the friendship between Edna and these two other people. To do so, he first sketches a line between Edna and Erica. The edge glyph reveals that they become friends at time 1 and remain friends for the rest of the time. He de-selects the edge glyph and then reveals a more interesting edge glyph between Edna and Jack. The edge glyph shows that they become friends at time 1, end their friendship momentarily at time 4 and then reconnect at the last time slice.

Focusing on the moment when they briefly end their friendship, he uses the edge slider to go directly to time 4 (Figure 5.9(B)). At time 4, he reveals the node glyph of Edna and notices that Edna has a new friend, Rosa, who was not previously noticed as part of the original group. Spencer decides to investigate Rosa's friendships, to see if she is friends with any other people in the original group.

He brings his attention to Rosa, by de-selecting Edna and revealing Rosa's node glyph that shows she was not present in the first two time slices, meaning that she didn't have any friends in the network. He activates the node time slider on Rosa's glyph to navigate directly to time 3, when she first emerges into the network and has only one friend, Irene. Continuing to move forward in time, at time 4, he notices that Rosa's degree increases as she befriends Jack and Edna, who are part of the original group, along with two new people: Jacob and Rob (Figure 5.9(C)). He wonders if Jacob or Rob are ever friends with Edna. He stops at time 4, hides Rosa's node glyph and then sketches an edge between Rob and Edna. He immediately sees they are never friends, as indicated by the dotted grey glyph. He does the same between Jacob and Edna, and discovers that they were friends, at two consecutive time slices.

He wonders if there was ever a time when Edna, Jacob and Rosa were ever all friends with each other, indicating the formation of a small clique. Leaving the revealed edge between Edna and Jacob, he adds another two edges between Jacob and Rosa, as well as, Edna and Rosa. The resulting aggregated edge glyphs reveal that they were all friends with each other once, at time 3 (Figure 5.9(D)). Spencer decides to further investigate the friendships of Jacob to see if he is friends with any other people in the original group. By combining Glidgets techniques, he was able to identify a group of friends within the network and investigate how friendships among the group vary over time. This lead to new discoveries, such as a small clique within the group and new members of the group that were added over time.

Chapter 6

Glidgets Evaluation

An exploratory evaluation was conducted to compare Glidgets to the regular time slider, when used to complete different types of tasks. In this chapter we discuss the design of our evaluation, our results and list some implications for the design of Glidgets and the evaluation.

6.1 Exploratory Evaluation

We performed a comparative, exploratory evaluation between Glidgets and the regular time slider, when used to complete tasks targeting low-level topological changes in a dynamic graph. Task completion time and error rate were measured for each task.

6.1.1 Task and Dataset Design

Our tasks were derived from previous experiments ([4], [6]) evaluating the readability of dynamic graphs. Related to the task taxonomy created by Bach et al. we created *temporal tasks*, requiring participants to locate a moment in time when an element, or set of elements had a certain temporal characteristic [6]. Our tasks focus on observing node degree changes and element presence/absence changes of both individual and sets of elements:

Node Degree (ND): Observe how a node's degree changes over time.

Node Presence (NP): Observe how the presence of a node changes over time.

Node Presence Set (NPS): Observe how the presence of three nodes changes over time.

Edge Presence (EP): Observe how the presence of an edge changes over time.

Edge Presence Set (EPS): Observe how the presence of three edges changes over time.

The tasks also relate to the questions that inspired the design of Glidgets (Section 5.1.1). We grouped the types of tasks into two sets: node-centric (ND, NP, NPS) and edge-centric (EP, EPS) (See Appendix B for all tasks). In order to simulate realistic data we altered an existing dataset, the Van De Bunt social network [56], to ensure each task had a unique answer. The same dataset was used in both technique conditions. In total, the graph consisted of 31 nodes, 111 edges, and 6 time slices. Nodes were labelled with numbers, according to the original dataset.

In summary, we used a 2 *technique* (Glidgets, regular time slider) x 2 *task set* (edge-centric, node-centric), within-participants design. The assignment order of *technique* and *task set* was counter-balanced across participants. In total, *technique* (2) x *task set* (9 node-centric + 5 edge-centric) x 8 participants = 224 trials were measured.

6.1.2 Procedure

Prior to the tasks, the experimenter explained the basics of a dynamic graph and graph properties, such as node degree, that were part of the tasks. Prior to each technique, participants engaged in a training phase where the experimenter demonstrated and explained each technique. The participant was then given as much time as needed to familiarise themselves with the technique. All participants were instructed to complete tasks as quickly and as accurately as possible. At the beginning of the task, the participants were provided with as much time as needed to read and understand the task description. The task description was presented on an iPad, with only one description visible at a time. Then, the participant pressed a “ready” button and the graph was displayed on the screen. Participants interacted with the graph using a stylus to solve the task. When a solution was found, participants spoke their answer aloud, which was recorded by the experimenter and pressed a “done" button. A blank screen was displayed in between tasks. At the beginning of each task, the time slider was re-set to the first time slice.

Task completion time was measured as the difference between the time when the graph was displayed (pressing the “ready" button) and the time when the task was completed (pressing the “done" button). Completion times were automatically logged by the system. The experimenter observed and recorded participant interactions while they solved the tasks. Participants could not re-do any task, however after pressing the “done" button, they could cancel their submission by pressing a “cancel” button, which displayed the graph and resumed the timer. In this case, task completion time was automatically adjusted by the system to account for the extra time.

Four questionnaires collecting subjective ratings were administered, each at the end of a task set for both techniques. The study session ended with a semi-structured interview, to collect feedback from participants regarding the Glidgets design and their experience using the glyphs.

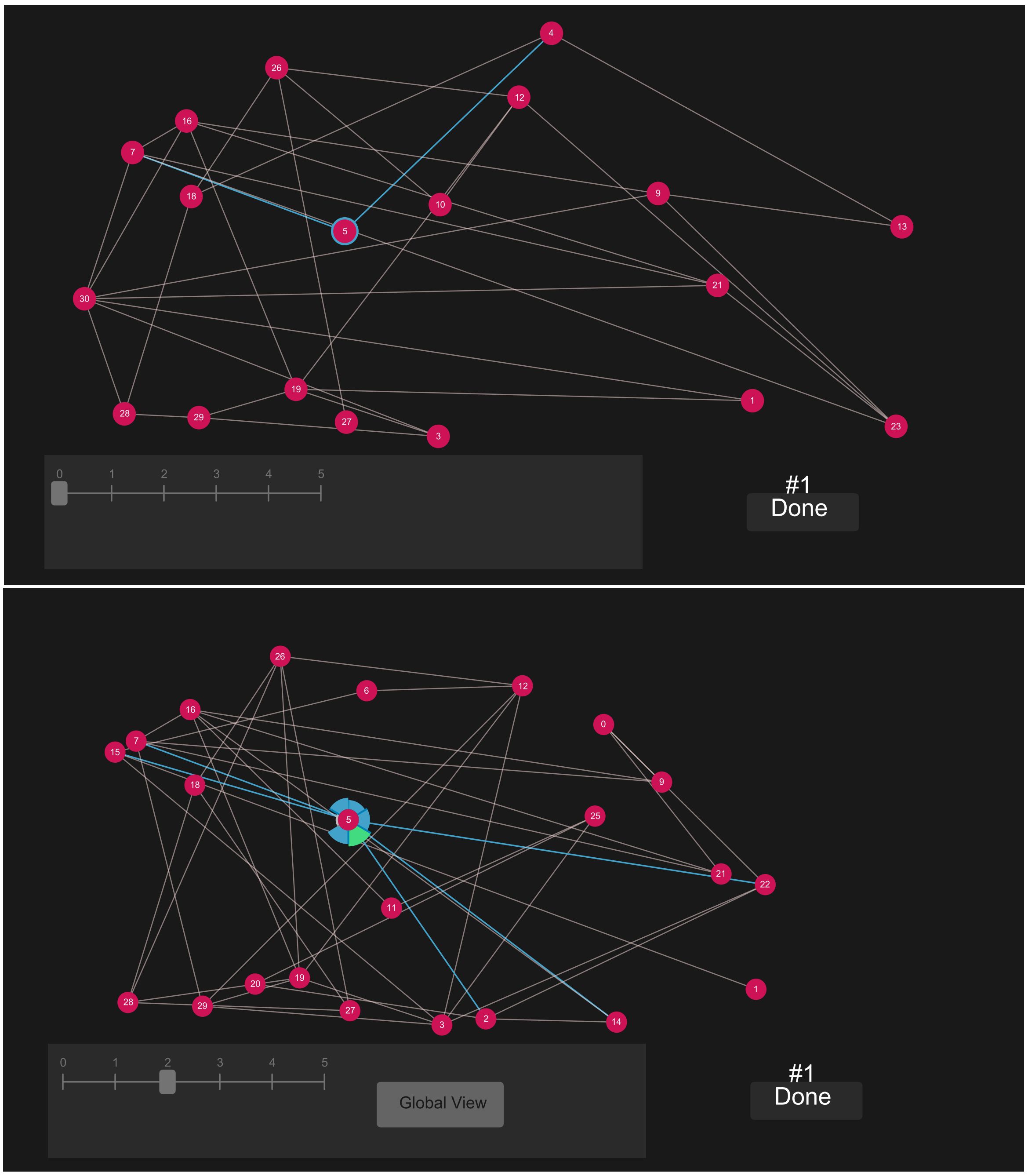

Fig. 6.1 The interfaces used in evaluation, for the two techniques: regular time slider (top) and Glidgets (bottom).

6.1.3 Interface Designs

We created different experimental interfaces for both the Glidgets and regular time slider technique conditions. In the experiment graph, the node positions were fixed using a "flip-book" style layout. The fading in and out of graph elements was timed using a cubic ease in and out function, and animation speed was user-controlled by dragging the time sliders. Below we describe the two types of technique interfaces used in the evaluation:

Glidgets The following Glidgets techniques were accessible: node and edge change glyphs, aggregated glyphs, embedded time sliders and the global view. A regular, horizontal time slider was provided, mainly for locating a nodes not present in the current view. We did not provide the edit mode, for re-positioning nodes.

Regular Time Slider A horizontal time slider was provided for navigating time. Participants were able to interactively select nodes. Selected nodes had a solid, blue halo drawn around them, and incident edges were highlighted in blue. Similar to the change glyph, the halos remained visible until deselected, even when the node disappeared. Nodes could be deselected by tapping the node again, or the background.

All interface colours were verified using VisCheck, to ensure they were discernible by colour blind users [20].

6.1.4 Pilot Test

During pilot testing before running the evaluation, we received some useful feedback and suggestions. As mentioned in Section 5.1.2 the global view was re-designed as an interactive temporary mode because participants found the view useful as a starting point and wanted to transition directly into the regular graph view. Originally, the edge of a glyph segment was highlighted orange, as a temporal indicator. Since the highlighted edge was in between two glyph segments, participants were unsure which segment represented the current time slice. In the current design, the entire segment is highlighted orange.

6.1.5 Experimental Setup

The task descriptions were displayed on an iPad. A Lenovo x220 tablet laptop with a 1366x768 screen resolution, running Windows, was used to run and display the graph. The laptop's display was mirrored on a second, larger monitor, for the experimenter to observe the participant's actions. Participants interacted with the graph to complete the tasks using a stylus, while seated, in a private laboratory with lower lighting conditions

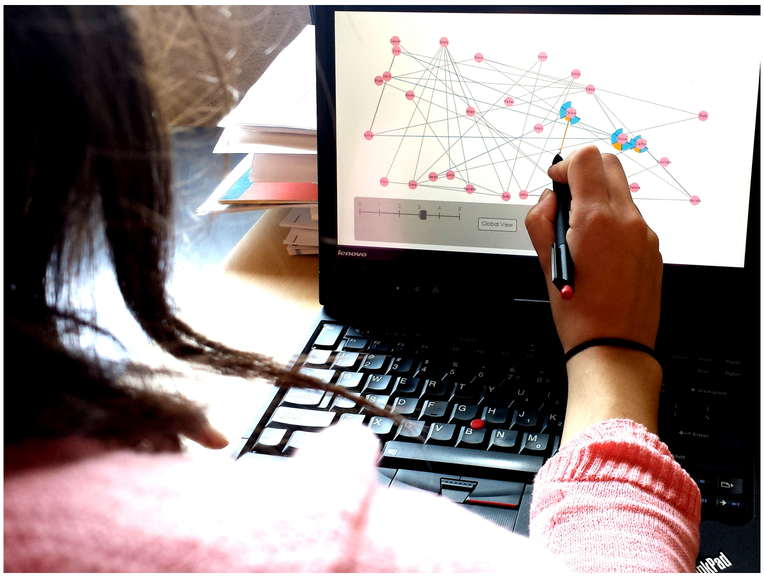

Fig. 6.2 Participants interacted with the graph using a pen, while seated.

(Figure 6.2). Participants were informed that they could take breaks, any time between tasks. On average, the study lasted around 1 hour. Participants received gift cards as compensation.

6.1.6 Participants

Eight participants (6 male and 2 female), aged between 20 to 30 years, were recruited from UOIT and surrounding area. All participants were right-handed and Glidgets was new to them.

6.2 Quantitative Results

For each task, participants performed differently because they used different combinations of techniques to solve the tasks. Not only did it take them time to plan which techniques to use, but, time was also spent switching between techniques when solving a task. This resulted in highly variable task completion times. Overall, the task completion time for Glidgets ($M = 44.3s$) was higher than the regular time slider ($M = 35.5s$). Error rates were low, and nearly uniformly distributed for each task. Overall, the regular time slider ($Errors = 12/112$) performed more accurately than Glidgets ($Errors = 18/112$). Due to the variability in our measures and small number of participants, no significant differences

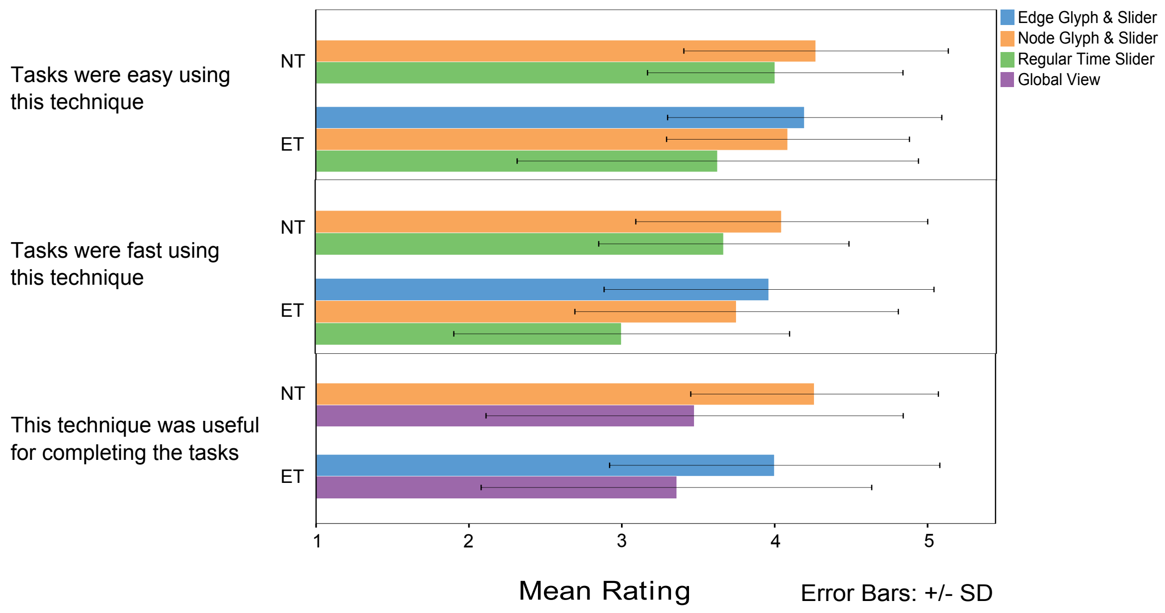

Fig. 6.3 Subjective ratings comparing the Glidgets techniques to the regular time slider, when used for completing node-centric (NT) and edge-centric tasks (ET). Ratings correspond to a 5-point Likert scale (1-Strongly Disagree to 5-Strongly Agree).

could be found in our quantitative measures.

6.3 Qualitative Results

In this section, we summarize our qualitative results gathered from subjective ratings and observations.

6.3.1 Subjective Feedback

Subjective ratings were measured after each task set, giving participants an opportunity to rate each Glidgets technique for the edge and node-centric tasks. If a participant did not use a certain technique to complete some of the tasks in the set, then they did not rate it, and were excluded from our mean ratings.

Overall, the node glyph and node slider were well received by participants. One of the participants expressed that the node glyph was “really representative of how the node changes over time.” Another participant stated that the node glyph and node time slider “highlight all the information that’s needed,” and another stated that they were “useful, easy and intuitive.” Subjective ratings revealed that, overall, completing node-centric tasks

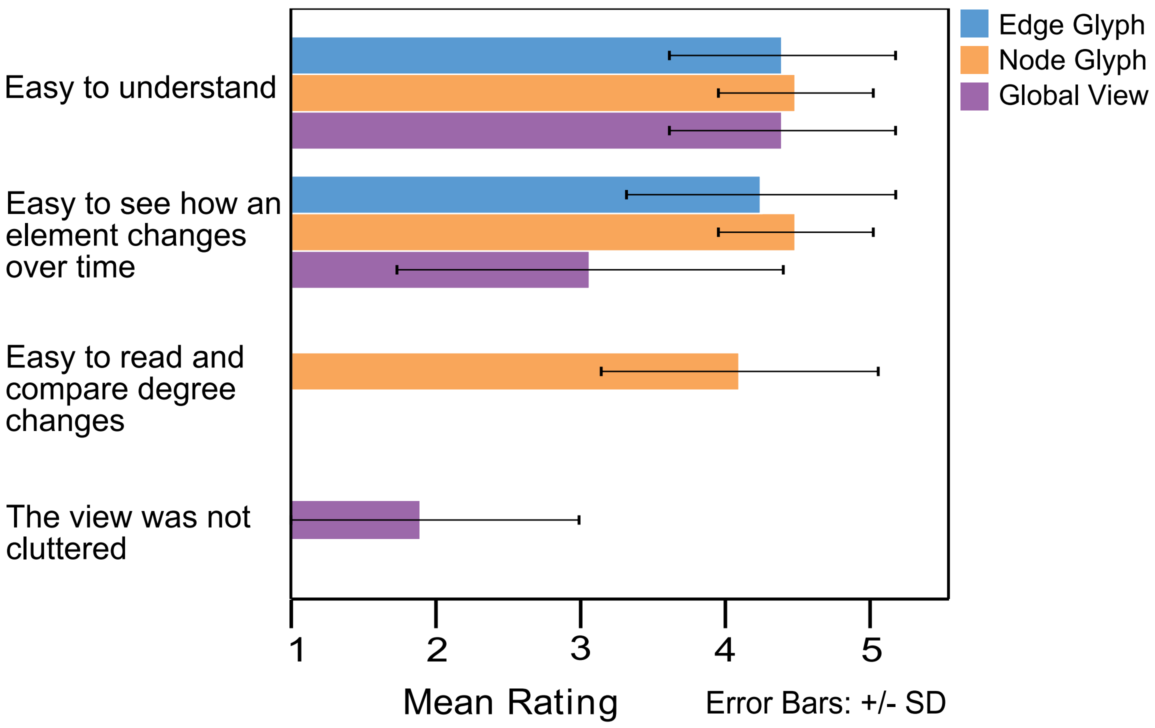

Fig. 6.4 Subjective ratings for the visual design of edge and node glyphs, and global view, when used to complete tasks. Ratings correspond to a 5-point Likert scale (1-Strongly Disagree to 5-Strongly Agree).

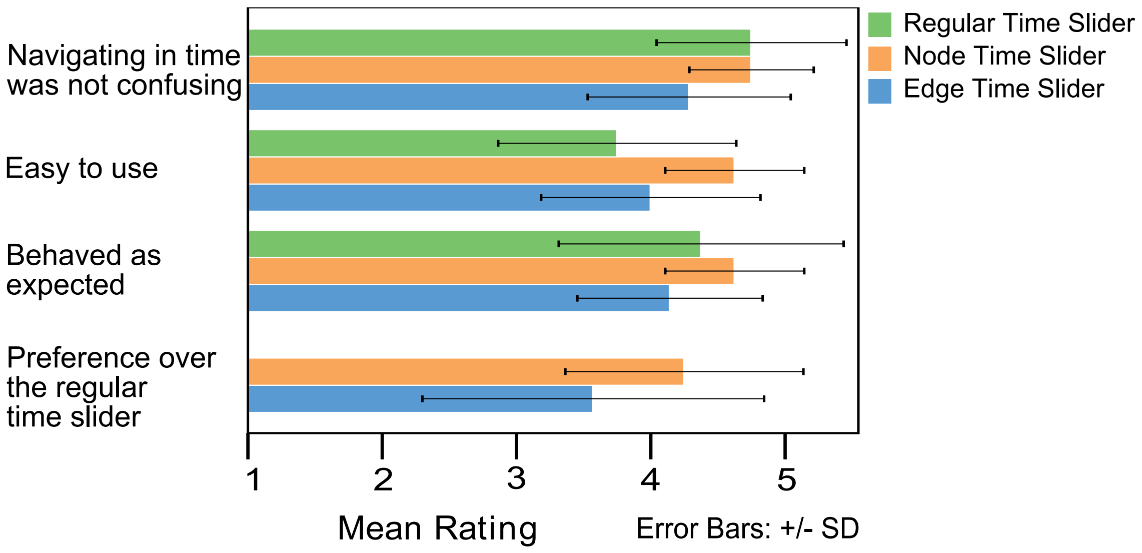

Fig. 6.5 Subjective ratings for the edge, node and regular time sliders when used to complete tasks. Ratings correspond to a 5-point Likert scale (1-Strongly Disagree to 5-Strongly Agree).

felt slightly faster and easier using the node slider (Figure 6.3: $Measy = 4.3$, $Mfast = 4.0$), compared to the regular time slider (Figure 6.3: $Measy = 4.0$, $Mfast = 3.7$). Additionally, edge-centric tasks felt faster and easier using the edge (Figure 6.3: $Measy = 4.2$, $Mfast = 4.0$) and node (Figure 6.3: $Measy = 4.1$, $Mfast = 3.8$) sliders, compared to the regular time slider (Figure 6.3: $Measy = 3.6$, $Mfast = 3.0$).

Six participants stated that they preferred using the node time slider over the edge time slider for completing the tasks, whereas two others thought that the choice of using either was task-dependent. One participant commented that sometimes using the regular time slider was faster than the other time sliders, but it required a lot of attention, and using the edge and node time sliders was more relaxing because the change information was displayed in the glyphs, while navigating through time by directly interacting with time sliders embedded in elements.

Related to the visual design of the glyphs, participants seemed to agree that the glyphs were easy to understand (Figure 6.4: $Medge = 4.3$, $Mnode = 4.4$) and that they easily show how an element changes over time (Figure 6.4: $Medge = 4.1$, $Mnode = 4.4$). Ratings for the usability of the embedded time sliders were mainly positive. Participants mainly agreed that both the node (Figure 6.5: $M = 4.6$) and edge (Figure 6.5: $M = 4.1$) time slider behaved as they expected. The node time slider received the strongest rating for ease of use (Figure 6.5: $M = 4.6$), followed by the edge slider (Figure 6.5: $M = 4.0$) and then the regular time slider (Figure 6.5: $M = 3.8$).

6.3.2 Use of Glidgets Techniques

Overall, few participants used only one Glidgets technique to solve a task; mainly combinations of techniques were used. These combinations varied for nearly every participant, in all the tasks. For each task, we counted the number of participants that used a technique at least once in solving a task. In doing so, we determined which techniques were used to solve a task, and how many participants used each technique to derive their solution (Figure 6.6). The use of glyphs was counted if a participant revealed the glyph, but did not use its slider.

Unexpectedly, some participants used the node glyph and slider to complete edge-centric tasks (See tasks EPS(12), EPS(13) and EPS(14) in Figure 6.6), however the edge glyphs and sliders were not used in node-centric tasks. While the node glyphs and sliders were designed to answer node questions, they can also be used indirectly to answer edge presence questions. However, the opposite is true in that the edge glyphs could be used to answer node presence questions. We suspect that participants might have been more

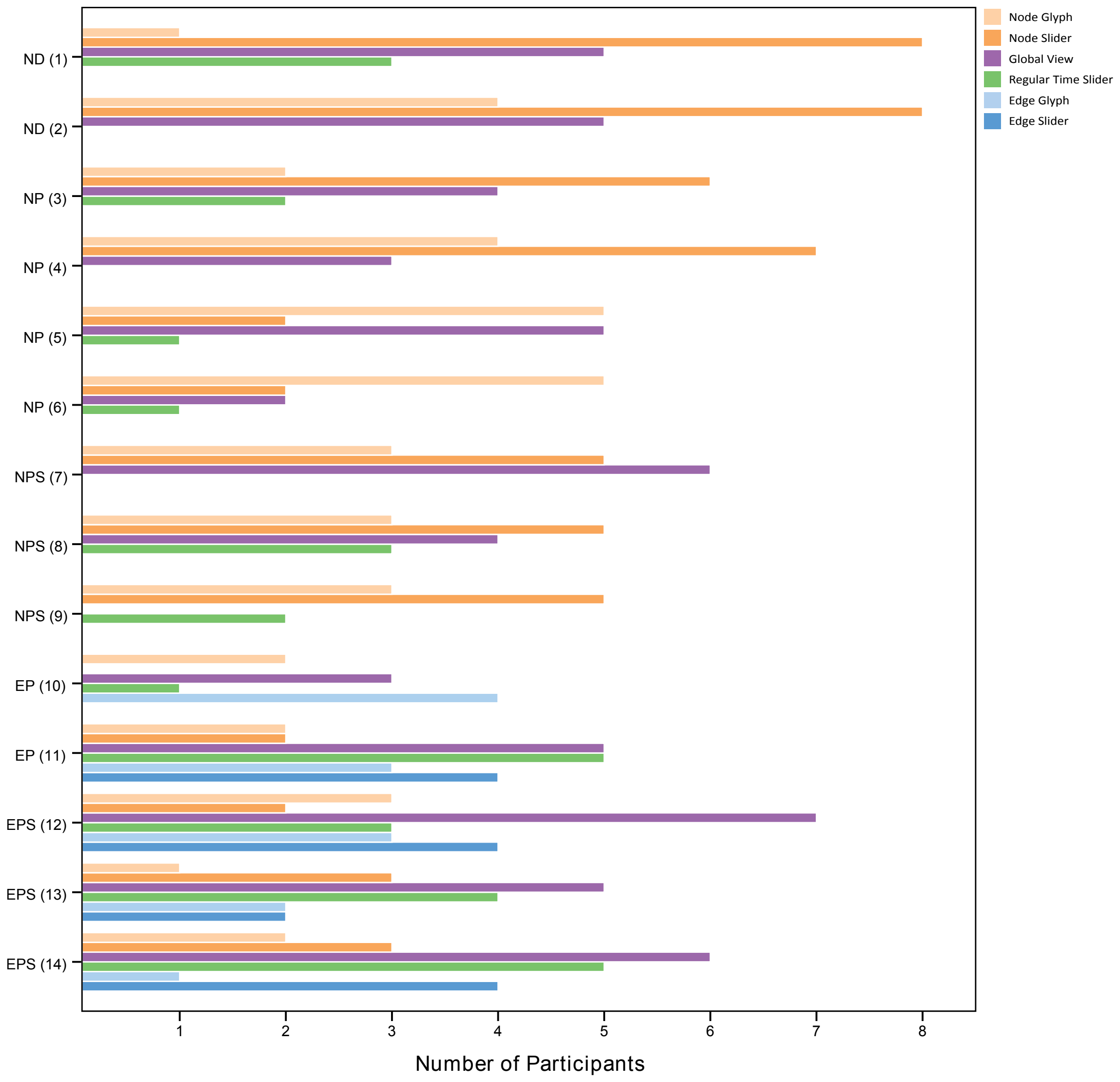

Fig. 6.6 The number of participants that used each technique at least once for solving the tasks. Each bar represents a count out of eight participants, however many participants used multiple techniques to solve a single task. The vertical axis is labelled as task type, followed by task number, in brackets.

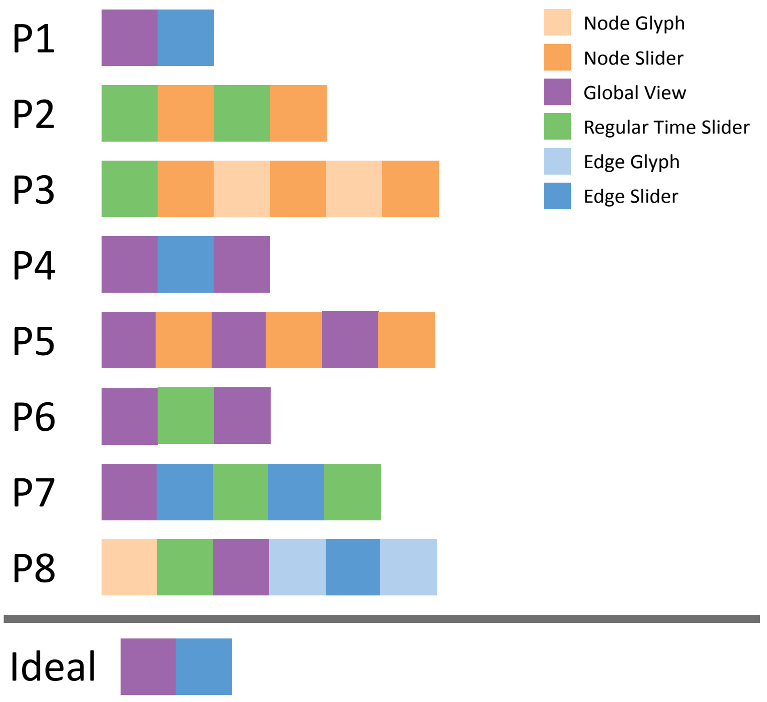

Fig. 6.7 The Glidgets techniques used by each participant for an EPS (14) task, ordered by time from left to right. No time information was recorded, therefore the coloured blocks differentiate the order of techniques and do not represent the duration of use. At the bottom, we present an ideal sequence of techniques to solve this task (global view, followed by edge time sliders).

comfortable using the node glyphs and sliders. One participant explained that they became comfortable with using the node glyph and slider, so it was more convenient to use them for solving all tasks.

Seven participants used the global view to solve at least one task. One participant described the view as "more like a starting point," implying that it was useful for locating nodes. Two participants used the view for almost every task, and were able to solve some tasks using only the global view. Seven participants used the regular time slider to solve at least one task, where the majority appeared to use it for either locating nodes involved in the tasks, or as alternative navigation technique alongside the embedded time sliders. The regular time slider was always used in conjunction with at least one other Glidgets technique.

Figure 6.7 provides a detailed representation of the different combinations of Glidgets techniques used by participants to solve task 14 ("After the first moment these edges appear together, which edge, out of edges A-B, A-C and A-D, is the first to disappear?"). Our ideal sequence of techniques to solve this task is: using the global view to select the edges, followed by dragging along an edge time slider to transition out of the global view and identify the disappearing edges. It is evident that all participants, except P1, deviated from the ideal sequence and created their own strategies to find a solution (Figure 6.7). Switching between embedded and regular time sliders (P7, P8 in Figure 6.7) may indicate a usability problem with the embedded sliders, as some participants reported that the edge slider was difficult to acquire. Additionally, P8 stated that they were confused by the automatic query aggregation of edge glyphs, for this task.

6.4 Technique Design Implications

Based on observations and feedback obtained from our exploratory evaluation we list some implications for the design of Glidgets:

User-controlled query aggregation: We observed that some participants expected multiple element selections to reveal change glyphs independently, and were confused by the automatic aggregation of the glyphs. Therefore, query aggregation should be provided as a separate mode activated by the user. Additionally, there were complaints about un-intended de-selections by accidentally tapping the background. A separate area such as a border surrounding the graph, could be allocated for de-selecting elements.

Show both relative and exact values: Some participants referred to the regular time slider when using the node or edge glyphs to solve tasks, despite the current time slice was displayed next to the pen. One participant suggested displaying all time slice labels along the glyphs. This reduces the simplicity and conciseness of the design, but does not require users to refer to the time slider as a navigation cue. The node glyphs portray relative node degree. During the node degree tasks, we observed some participants using the glyph to locate highest or lowest degree, but then they would verify their answer by counting the edges. This may indicate a need for access to both relative and exact values of change metrics. The node degree values could be overlaid on the glyph when the user wishes to see exact amounts, but hidden otherwise.

Perceptible disappearing edges: Disappearing edges were not very perceptible when using the edge time slider because of the occluding edge glyph, particularly when multiple edges were selected. Thus, disappearing edges should be more visually prominent. Other than experimenting with different colours, we could consider one of our original designs where the glyph surrounds the edge, as a border. This would resolve the occlusion, but require the glyph to be thicker.

Easily comparable glyphs: Three participants stated that since the arrows along the edge glyph can be oriented in opposing directions, making comparisons between multiple glyphs was difficult. This is an interesting finding, as since the glyphs are automatically aggregated, all glyphs of the selected edges are the same. This further supports that participants did not understand or detect the automatic aggregation.

Nevertheless, the readability of multiple edges was not considered in our design. Currently, the edge glyph is drawn to ensure the arrows point toward the right-most node. We assumed that this would improve the readbility of edges by matching the glyph's time line layout to the natural, left-to-right reading order of the regular time slider. For aggregated edge glyphs, we could try to match the arrow directions such that edges point towards a similar direction. However, a more substantial solution might be reconsidering the aggregated glyph design. For instance, multiple glyphs could be combined in a single glyph, aligning the time segments for easier comparison.

Effective overview of changes: When using the global view to search for elements of interest, many participants requested a faster method of locating the elements. In particular, one suggestion was adding a search function. Additionally, some participants found the fading effect made it difficult to search for other elements, suggesting that the transparency should be an adjustable parameter. To improve the global view's capabilities, we could consider generating interesting initial views of the graph (e.g., all nodes that are always present over time) or, provide sorting functions to order nodes (e.g., alphabetically, overall degree amount).

6.5 Study Design Implications

Below we list some notable issues with our study design that could be used to inform the design of future evaluations:

Feature overload: Since our evaluation was exploratory, our initial goal was to determine if and how participants used the different Glidgets techniques to solve tasks. However, five out of eight participants commented that they felt overwhelmed initially with the multitude of features offered by Glidgets and expressed that there is a steep learning curve. In future evaluations, it may be beneficial to isolate the Glidgets techniques. For instance, providing only the node glyph and slider to solve node-centric tasks, and directly comparing its performance to the regular time slider. While this imposes constraints on task completion strategies, we can ensure that completion times are directly measured from the use of the same techniques.

Overhead of the search task: We frequently observed participants spending time searching the graph for nodes required to complete a task, and this search task was time-consuming. In future evaluations, node(s) required to solve a task should be discernible from the rest of the nodes (e.g., automatically highlight them).

Inadequate amount of training: It would be optimal to have participants practice each task to ensure they do not encounter a task they have not previously practiced.

Potential learning effects: In both technique conditions, the same dataset was used. Even though each task had a unique answer, while not tested, there is still a possibility of learning effects to impact results, especially since the node positions were fixed. Future evaluations should use different datasets in each condition, or randomize the layout.

Other comparison techniques: The regular time slider was at a slight disadvantage because no change information was displayed in the highlights. A fairer comparison to Glidgets would be presenting the same change information (element presence and node degree) in a separate view, perhaps using a small line chart. Additionally, the small multiples technique with difference highlighting could be added as another comparison technique.

Considering the effect of time line size: Six time points were selected for the evaluation, because that was the size of our original dataset, and we wanted to keep the study under a reasonable length. However, based on our observations, using more time points may produce different performance results. Participant actions to complete tasks demonstrated an example of the GOMS (Goals, Operators, Methods and Selection rules) model [15]. Participants were selecting Glidgets components to develop a method that would help them reach their goal (i.e., solving the task). Here, participants were planning and then executing their task solution strategies, whereas when using the time slider, a single operator, there was only execution. Due to this lack of planning and the mental effort required to manually track and remember element changes, it may be increasingly difficult to complete tasks using only the time slider, as the time line size increases.

Chapter 7

Conclusion

Temporal navigation controls are traditionally employed as separate widgets, that are typically distant from the changing visualization objects. Manipulating separate controls to explore time is incompetent for tracking and understanding changes of individual data objects, because it requires the user to deviate their focus away from the object of interest. In our approach, the navigation control and visualization of object changes are embedded in a data object, harmonizing temporal navigation and analysis of changes.

In this chapter we summarize our contributions, discuss the scalability of our object-centric techniques and present ideas for future work.

7.1 Contributions

Our main contribution is object-centric temporal navigation, a technique for exploring time by embedding a time slider visualizing temporal changes in the visual object of interest. To illustrate object-centric temporal navigation, we designed two techniques: DimpVis, for navigating information visualizations and Glidgets, for navigating dynamic graphs. For both techniques, we presented results from evaluations comparing our techniques to existing temporal navigation methods.

7.2 Limitations

While not investigated in detail, we discuss the scalability and limitations of our techniques, DimpVis and Glidgets.

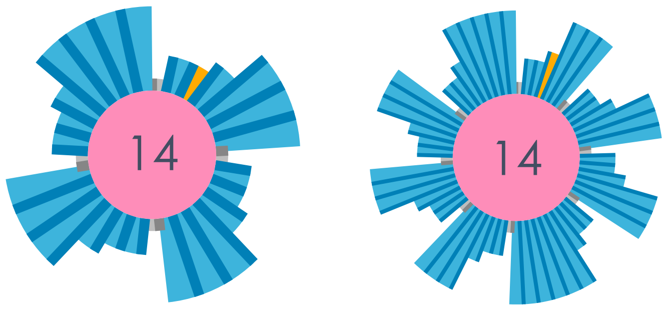

Fig. 7.1 A node glyph for 30 (left) and 60 (right) time slices.

7.2.1 Scalability of DimpVis

Larger datasets generally make trend detection and tracking items of interest more difficult in animated plots, and this may be true for the scalability of DimpVis too. Techniques for filtering or focusing on subsets of data items of interest could be used, such as lenses or highlights. DimpVis may encounter interaction usability challenges in dense regions of a visualization. For instance, if a scatter plot is too dense to perceive individual points, then directly interacting with them may not be feasible without filtering the dataset. Also, if the time scale of a dataset is large, the hint path will become long and potentially cluttered. A cluttered hint path may require aesthetic (e.g., thinner lines, aggregating time points) or functional (e.g., a scrolling hint path) enhancements.

7.2.2 Scalability of Glidgets

In general, representing and navigating dynamic graphs is challenging due to known scalability issues that arise when the graph grows in size. Similarily, Glidgets may suffer from these known scalability issues. As the time line increases, the change glyphs become less readable, as it is difficult to discern and track time slices (Figure 7.1). Similarily, the edge change glyph requires a minimum amount of space between nodes for it to be legible. For this reason, nodes can be interactively re-positioned in Glidgets. In both cases the glyph could be temporarily enlarged during interaction, to see its details. Furthermore, time labels may need to be added to improve readability of the glyphs. However, this could clutter the view. As the number of graph elements increases, the global view becomes more cluttered. As suggested by participants, a search function may help the user locate elements, or sub-trees could be compressed into a single visualization showing an overview of the element changes in the sub-tree (e.g., [1]).

Glidgets eliminates the effect of node movement by fixing the positions of nodes. This limits the versatility of our technique because certain types of graph datasets (e.g., document similarity networks) use node position to convey a data attribute. We could experiment with different layout algorithms optimizing node and edge position to lower the effect of node movement during temporal navigation. Lastly, the parameters of the force-directed algorithm did not produce an ideal layout and consequentely node positions appeared random. Better tuning of the algorithm parameters should be investigated.

7.3 Conclusion

Popular temporal navigation controls for exploring visualizations tend to be disjoint from the changing data items. Therefore, the user must shift their focus between observing an object of interest to see visualized temporal changes, and using a separate control, such as a time slider, to see temporal location. We introduced object-centric temporal navigation that narrows the gap between the user and visual items of interest, by directly manipulating items along their visual trajectory of temporal changes. To illustrate object-centric temporal navigation, we designed and evaluated two interaction techniques: DimpVis, for exploring varying information visualizations, and Glidgets, for exploring dynamic graphs. We implemented DimpVis for touch interaction with bars (bar chart), points (scatter plot), coloured cells (heat map) and angular segments (pie chart), and Glidgets as a pen-based interface for exploring dynamic graphs.

While DimpVis did not significantly out-perform the traditional time slider in our comparative evaluation, DimpVis for the scatter plot was subjectively preferred by participants overall and was significantly faster than the small multiples technique. Participant feedback and results from the evaluation motivate us to further explore and evaluate different design components, such as alternative hint path designs and navigation methods. Our study results also indicated the need for multiple types of hint paths, which encouraged us to add a flashlight hint path to the scatter plot and bar chart prototypes.

In our exploratory evaluation, Glidgets was generally subjectively preferred by participants for solving tasks, compared to the regular time slider. While the quantitative measures were not in favour of Glidgets, we were able to gain a better understanding of how users might combine multiple Glidgets techniques to answer different types of analytical questions.

We intended object-centric temporal navigation as a navigation technique complimentary to existing techniques, such as the time slider. By integrating our technique time-

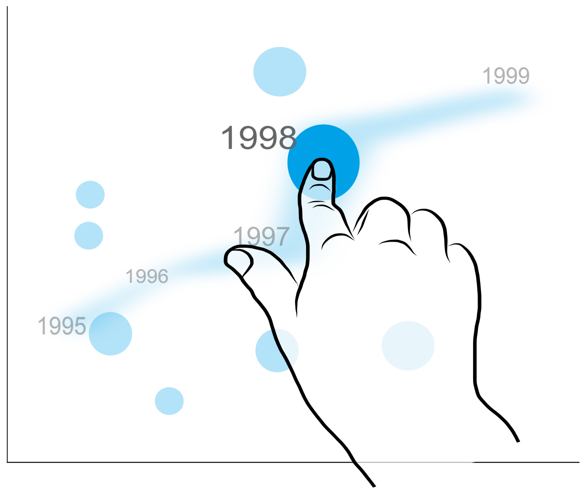

Fig. 7.2 An illustration of a time line hint path encoding changing position and size. As the user drags along the path, the point changes in position and size.

varying visualization systems can offer methods for exploring data object changes and object-focused temporal navigation, tasks that existing techniques were not designed to support, and improve interface flexiblity [38].

7.4 Future Work

Both DimpVis and Glidgets are initial examples of how object-centric temporal navigation can be applied to different types of information visualizations. We have some ideas for future work related to expanding the capabilities of our techniques, considering design alternatives for specific components of our design and additional evaluations for assessing different aspects of our techniques.

7.4.1 DimpVis

Multi-attribute hint paths and manipulation:

The generality of DimpVis can be expanded by applying it other types of dynamic information visualizations, with different types of changing visual variables. For instance, items in a time-varying bubble chart (e.g., [46]) often have two changing visual variables: size and position. This would require both a multi-attribute hint path and potentially more complex interaction techniques for manipulating data items, such as multi-touch gestures. For instance, a time line hint path could show how the point's position changes over time,

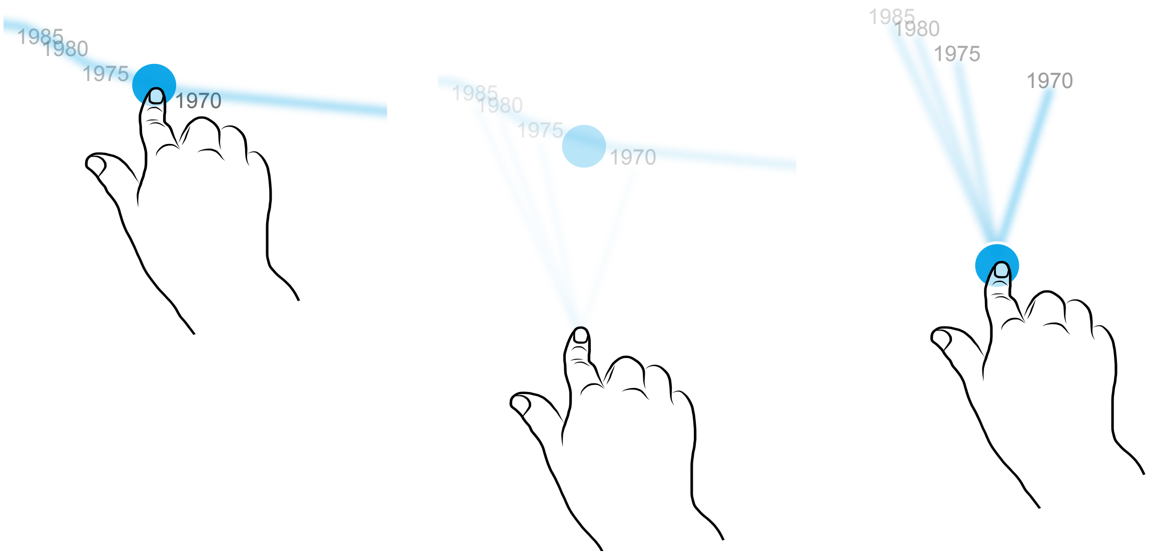

Fig. 7.3 A sketch showing the transition from a time line to flashlight hint path. Starting with the time line hint path (left), as the finger deviates from the path, the flashlight path gradually appears and the time line fades away (middle). Then, the point snaps to meet the finger (right).

with the added size attribute (e.g., encoded in the thickness of the different line segments, Figure 7.2).

Additionally, one might be interested in how size and position of a point change together or independently. A flashlight hint path could be designed to query changing visual variables simultaneously and independently. For example, to answer the question, "When is this point the largest?", two fingers could be used to drag the point outwards, enlarging it's size without changing it's position. When the dragging ends, the point can be transformed to the nearest size on the flashlight path, and then translated to it's position at that time. Querying both variables simultaneously may involve a sequence of gestures, where the user first drags the point to the desired position and then manipulates the size of the point to see if and when it had a certain size, at that position.

Alternative hint path design and navigation: Since the hint path has a direct effect on the type of temporal navigation, exploring other hint path designs would expand the flexibility of DimpVis. Currently, the time line and flashlight hint paths are separate interface modes. While not implemented, we have given some thought to a combined hint path enabling seamless transitioning between the two path types, supporting both direct queries and temporal trend analysis. For instance, the user begins with the time line hint path and if they deviate from the path, the hint path could gradually transform into a flashlight hint path (Figure 7.3).

Enabling multi-item comparison: While our DimpVis design was mainly data object-focused, some participants suggested better support for comparing the hint paths of multiple items. This may involve simply providing the ability to reveal multiple hint paths, or creating more sophisticated querying capabilities, such as presenting all hint paths that have a similar characteristics to a selected path.

Application to interactive storytelling Inspired by our evaluation, we would like to study the usefulness of DimpVis for storytelling or presentations of dynamic data visualizations, as hinted at by our subjective feedback. For instance, when a presenter wishes to draw emphasis to a data item's trend they can show the hint path and then navigate to time points when the item has an interesting value, while maintaining focus on the item.

Additional evaluations: Although we initially argued that DimpVis was suitable for touch input, we would be interested in comparing touch and mouse input. Also, our evaluation is limited to bar charts and scatter plots. Therefore performing additional evaluations with other types of visualizations may reveal other usability issues with DimpVis, when applied to other types of visual variables. Lastly, we would like further comparatively evaluate our flashlight and time line hint path designs.

7.4.2 Glidgets

Stronger comparative evaluation: We plan to perform another comparative evaluation with Glidgets and the regular time slider, incorporating most of the changes discussed in Section 6.5. Additionally, we would like to measure the effect of time line size on the Glidgets and time slider techniques, by using a larger number of time points in the dataset.

More elaborate queries: We would like to expand the querying capability to answer other types of questions, such as: "When do all selected nodes disappear together?". Though we focused only on visualizing element presence and node degree changes, other temporal changes could be integrated with the current glyph design (e.g., edge weight changes over time, shown by increasing thickness of glyph segments), or added as another type of glyph (e.g., node position changes could be visualized as trajectories).

Faster selection techniques: In order to select multiple elements, each element must be selected individually. Alternative selection techniques for multiple elements, such as a lasso selection for multiple nodes, or drawing a continuous path through nodes for multiple edges, may be more efficient techniques for selecting many elements at once.

Scaling the glyph design: Some limitations of our change glyphs is that they may not scale well to larger time lines and it is difficult to compare multiple glyphs. Exploring additional designs that are more scalable is necessary to expand the generality of Glidgets.

For instance, the node glyph could be layered to show aggregations of time and explore different levels of time. To support comparison, glyphs could be combined into a single glyph that presents all glyph segments encoding changes near each other, for easy comparison. Alternatively, we could offer more layout flexibility to overcome orientation constraints. For instance, to support comparison tasks, node glyphs could be rolled out into a straight line.

References

[1] Abello, J., Hadlak, S., Schumann, H., and Schulz, H. (2013). A modular degree-of-interest specification for the visual analysis of large dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):337–350.

[2] Ahn, J.-w., Taieb-Maimon, M., Sopan, A., Plaisant, C., and Shneiderman, B. (2011). Temporal visualization of social network dynamics: prototypes for nation of neighbors. In *Social Computing, Behavioral-Cultural Modeling and Prediction*, pages 309–316. Springer.

[3] Amar, R., Eagan, J., and Stasko, J. (2005). Low-level components of analytic activity in information visualization. In *IEEE Transactions on Visualization and Computer Graphics*, pages 111–117. IEEE.

[4] Archambault, D., Purchase, H. C., and Pinaud, B. (2011a). Animation, small multiples, and the effect of mental map preservation in dynamic graphs. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):539–552.

[5] Archambault, D., Purchase, H. C., and Pinaud, B. (2011b). Difference map readability for dynamic graphs. In *Graph drawing*, pages 50–61. Springer.

[6] Bach, B., Pietriga, E., and Fekete, J. (2013). Graphdiaries: Animated transitions and temporal navigation for dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 20(5):740–754.

[7] Baudel, T. (2006). From information visualization to direct manipulation: Extending a generic visualization framework for the interactive editing of large datasets. In *Proc. of the ACM Symposium on User Interface Software and Technology*, pages 67–76. ACM.

[8] Baur, D., Lee, B., and Carpendale, S. (2012). Touchwave: kinetic multi-touch manipulation for hierarchical stacked graphs. In *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces*, pages 255–264. ACM.

[9] Beaudouin-Lafon, M. (2000). Instrumental interaction: An interaction model for designing post-wimp user interfaces. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 446–453. ACM.

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

[11] Brandes, U. and Corman, S. R. (2003). Visual unrolling of network evolution and the analysis of dynamic discourse. *Information Visualization*, 2(1):40–50.

[12] Brewer, C., Harrower, M., Ben, S., Woodruff, A., and Heyman, D. (2002). Color brewer 2.0: Color advice for cartography.

[13] Brown, E. T., Liu, J., Brodley, C. E., and Chang, R. (2012). Dis-function: Learning distance functions interactively. In *Proc. of IEEE Visual Analytics Science and Technology (VAST)*, pages 83–92. IEEE.

[14] Burch, M., Vehlow, C., Beck, F., Diehl, S., and Weiskopf, D. (2011). Parallel edge splatting for scalable dynamic graph visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2344–2353.

[15] Card, S. K., Newell, A., and Moran, T. P. (1983). *The Psychology of Human-computer Interaction.* L. Erlbaum Associates Inc.

[16] Coffey, D., Lin, C.-L., Erdman, A. G., and Keefe, D. F. (2013). Design by dragging: An interface for creative forward and inverse design with simulation ensembles. *IEEE Transactiosn on Visualization and Computer Graphics*, 19(12):2783–2791.

[17] Coleman, M. K. and Parker, D. S. (1996). Aesthetics-based graph layout for human consumption. *Software: Practice and Experience*, 26(12).

[18] Cui, W., Wang, X., Liu, S., Riche, N. H., Madhyastha, T. M., Ma, K.-L., and Guo, B. (2014). Let it flow: a static method for exploring dynamic graphs. In *Proc. of IEEE Pacific Visualization.*

[19] Dork, M., Carpendale, S., Collins, C., and Williamson, C. (2008). VisGets: Coordinated visualizations for web-based information exploration and discovery. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1205–1212.

[20] Dougherty, B. and Wade, A. (2008). Vischeck plugin for imagej. http://www.vischeck.com/downloads/.

[21] Dourish, P. (2001). *Where the Action Is: The Foundations of Embodied Interaction.* MIT Press.

[22] Dragicevic, P., Ramos, G., Bibliowitcz, J., Nowrouzezahrai, D., Balakrishnan, R., and Singh, K. (2008). Video browsing by direct manipulation. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, page 237, New York, New York, USA. ACM Press.

[23] Elmqvist, N., Dragicevic, P., and Fekete, J.-D. (2008). Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1539–1148.

[24] Elmqvist, N., Moere, A. V., Jetter, H.-C., Cernea, D., Reiterer, H., and Jankun-Kelly, T. (2011). Fluid interaction for information visualization. *Information Visualization*, 10(4):327–340.

[25] Endert, A., Bradel, L., and North, C. (2013). Beyond control panels: Direct manipulation for visual analytics. *IEEE Computer Graphics and Applications*, 33(4):6–13.

[26] Endert, A., Fiaux, P., and North, C. (2012). Semantic interaction for sensemaking: inferring analytical reasoning for model steering. *IEEE Transactions onVisualization and Computer Graphics*, 18(12):2879–2888.

[27] Farrugia, M., Hurley, N., and Quigley, A. (2011). Exploring temporal ego networks using small multiples and tree-ring layouts. In *ACHI 2011, The Fourth International Conference on Advances in Computer-Human Interactions*, pages 79–88.

[28] Federico, P., Aigner, W., Miksch, S., Windhager, F., and Zenk, L. (2011). A visual analytics approach to dynamic social networks. In *Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies*, page 47. ACM.

[29] Fisher, D., Nelson, T., and O'Madadhain, J. (2010). Jung: Java universal network/graph framework. http://www.ccs.neu.edu/home/amislove/twittermood/.

[30] Goth, G. (2011). Brave NUI world. *Communications of the ACM*, 54(12):117–119.

[31] Grossman, T., Matejka, J., and Fitzmaurice, G. (2010). Chronicle: Capture, exploration, and playback of document workflow histories. In *Proc. of the ACM Symposium on User Interface Software and Technology*, pages 143–152. ACM.

[32] Karrer, T., Weiss, M., Lee, E., and Borchers, J. (2008). DRAGON: A direct manipulation interface for frame-accurate in-scene video navigation. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 247–250. ACM.

[33] Karrer, T., Wittenhagen, M., and Borchers, J. (2012). Draglocks: Handling temporal ambiguities in direct manipulation video navigation. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 623–626. ACM.

[34] Kriglstein, S., Pohl, M., and Stachl, C. (2012). Animation for time-oriented data: An overview of empirical research. In *Proc. of the 16th International Conference on Information Visualisation (IV)*, pages 30–35. IEEE.

[35] Lee, B., Isenberg, P., Riche, N. H., and Carpendale, S. (2012). Beyond mouse and keyboard: Expanding design considerations for information visualization interactions. *IEEE Transactions on Visualization and Computer Graphics*, 18(12).

[36] Moody, J., McFarland, D., and Bender-deMoll, S. (2005). Dynamic network visualization1. *American Journal of Sociology*, 110(4):1206–1241.

[37] Moscovich, T., Chevalier, F., Henry, N., Pietriga, E., and Fekete, J.-D. (2009). Topology-aware navigation in large networks. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2319–2328. ACM.

[38] Nielsen, J. (1995). 10 usability heuristics for user interface design. http://www.nngroup.com/articles/ten-usability-heuristics/.

[39] Nightingale, F. (1858). Diagram of the causes of mortality in the army in the east. http://en.wikipedia.org/wiki/File:Nightingale-mortality.jpg.

[40] Perin, C., Vuillemot, R., and Fekete, J.-D. (2014). À table!: Improving temporal navigation in soccer ranking tables. In *Proc. of SIGCHI Conference on Human Factors in Computing Systems.*

[41] Pike, W. A., Stasko, J., Chang, R., and O'Connell, T. A. (2009). The science of interaction. *Information Visualization*, 8(4):263–274.

[42] Pohl, M., Reitz, F., and Birke, P. (2008). As time goes by: integrated visualization and analysis of dynamic networks. In *Proceedings of the working conference on Advanced visual interfaces*, pages 372–375. ACM.

[43] Reas, C. and Fry, B. (2007). *Processing: A Programming Handbook for Visual Designers and Artists.* MIT Press.

[44] Rind, A., Aigner, W., Miksch, S., Wiltner, S., Pohl, M., Drexler, F., Neubauer, B., and Suchy, N. (2011). Visually exploring multivariate trends in patient cohorts using animated scatter plots. In *Ergonomics and Health Aspects of Work with Computers*, pages 139–148. Springer.

[45] Robertson, G., Fernandez, R., Fisher, D., Lee, B., and Stasko, J. (2008). Effectiveness of animation in trend visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1325–1332.

[46] Rosling, H. (2007). Gapminder trendalyzer. http://www.gapminder.org.

[47] Rufiange, S. and McGuffin, M. J. (2013). Diffani: Visualizing dynamic graphs with a hybrid of difference maps and animation. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2556–2565.

[48] Saffrey, P. and Purchase, H. (2008). The mental map versus static aesthetic compromise in dynamic graphs: a user study. In *Proceedings of the ninth conference on Australasian user interface-Volume 76*, pages 85–93. Australian Computer Society, Inc.

[49] Sallaberry, A., Muelder, C., and Ma, K.-L. (2013). Clustering, visualizing, and navigating for large dynamic graphs. In *Graph Drawing*, pages 487–498. Springer.

[50] Santosa, S., Chevalier, F., Balakrishnan, R., and Singh, K. (2013). Direct space-time trajectory control for visual media editing. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1149–1158. ACM.

[51] Saraiya, P., Lee, P., and North, C. (2005). Visualization of graphs with associated timeseries data. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, pages 225–232. IEEE.

[52] Schreck, T., Tekušová, T., Kohlhammer, J., and Fellner, D. (2007). Trajectory-based visual analysis of large financial time series data. *ACM SIGKDD Explorations Newsletter*, 9(2):30–37.

[53] Shneiderman, B. (1997). Direct manipulation for comprehensible, predictable and controllable user interfaces. In *Proc. of the 2nd International Conference on Intelligent User Interfaces*, pages 33–39. ACM.

[54] Tominski, C., Schumann, H., Adrienko, G., and Adrienko, N. (2012). Stacking-based visualization of trajectory attribute data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2565–2574.

[55] Tufte, E. R. (1990). *Envisioning Information.* Graphics Press, Cheshire, USA.

[56] Van Duijn, M. A., Zeggelink, E. P., Huisman, M., Stokman, F. N., and Wasseur, F. W. (2003). Evolution of sociology freshmen into a friendship network. *Journal of Mathematical Sociology*, 27(2-3):153–191.

[57] Wolter, M., Hentschel, B., Tedjo-Palczynski, I., and Kuhlen, T. (2009). A direct manipulation interface for time navigation in scientific visualizations. *2009 IEEE Symposium on 3D User Interfaces*, pages 11–18.

[58] Yi, J. S., ah Kang, Y., Stasko, J. T., and Jacko, J. A. (2007). Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231.

[59] Zaman, L., Kalra, A., and Stuerzlinger, W. (2011). The effect of animation, dual view, difference layers, and relative re-layout in hierarchical diagram differencing. In *Proceedings of Graphics Interface 2011*, pages 183–190. Canadian Human-Computer Communications Society.

[60] Zhao, J., Chevalier, F., Pietriga, E., and Balakrishnan, R. (2011). Exploratory analysis of time-series with ChronoLenses. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2422–2431.

Appendix A

DimpVis Evaluation Tasks

Table A.1 The versions of each task type created for the scatter plot evaluation. 'A' and 'B' refer to labelled points, and 'X' and 'Y' represent scatter plot point values.

Task Type	Task Versions
Retrieve value (RV)	When is A's age < X and height = Y feet?
	When is A's age < X and height = Y feet?
	When is A's age = X and height = Y feet?
Comparison (CO)	When is A's age and height greater than B's?
	When is A's age and height less than B's?
	When is A's age and height equal to B's?
Characterize Distribution (CD)	After the age and height of A have been increasing, find the first year when they are both decreasing.
	After the age and height of A have been decreasing, find the first year when they are both increasing.
Outlier Detection (OD)	Find the first year when A is moving in the opposite direction of the other points.

Table A.2 The versions of each task type created for the bar chart evaluation. ’A’ and ’B’ refer to labelled bars, and ’X’ represents a bar chart height value.

Task Type	Task Versions
Retrieve value (RV)	When is A’s height = X?
	When is A’s height < X?
	When is A’s height > X?
Comparison (CO)	When is A’s height greater than B’s?
	When is A’s height less than B’s?
	When is A’s height equal to B’s?
Characterize Distribution (CD)	After the height of A has been increasing, find the first year when it is decreasing.
	After the height of A has been decreasing, find the first year when it is increasing.

Appendix B

Glidgets Evaluation Tasks

Table B.1 The different types of tasks used in our exploratory evaluation, this set of tasks was repeated for both *technique* conditions. Node-centric tasks are listed above the double line, and edge-centric tasks are below.

Task Type	Task Description
Node Degree (ND)	When node A is present, when is its degree the lowest?
	When node A is present, when is its degree the lowest?
Node Presence (NP)	Find the first moment when node A disappears.
	Find the first moment when node A reappears.
	After its first appearance, how many times does node A disappear?
	After its first disappearance, how many times does node A reappear?
Node Presence Set (NPS)	When do all of node A, B and C appear together?
	Find the first moment when none of node A, B and C are present.
	After the first moment they all appear together, which node, out of A, B, and C, is the first to disappear?
Edge Presence (EP)	Are nodes A and B ever connected?
	After the first moment they are connected, find the first moment when nodes A and B disconnect.
Edge Presence Set (EPS)	When do all of the edges A-B, A-C and A-D appear together?
	After the first moment these edges appear together, when do all of the edges A-B, A-C and A-D disappear together?
	After the first moment these edges appear together, which edge, out of edges A-B, A-C and A-D, is the first to disappear?