Visual Links across Applications

Manuela Waldner* Werner Puff† Alexander Lex‡ Marc Streit§ Dieter Schmalstieg¶

Institute for Computer Graphics and Vision
Graz University of Technology
Austria

Abstract

The tasks carried out by modern information workers become increasingly complex and time-consuming. They often require to evaluate, interpret, and compare information from different sources presented in multiple application windows. With large, high resolution displays, multiple application windows can be arranged in a way so that a large amount of information is visible simultaneously. However, individual application windows' contents and visual representations are isolated and relations between information items contained in these windows are not explicit. Thus, relating and comparing information across applications has to be executed manually by the user, which is a tedious and error-prone task.

In this paper we present visual links connecting related pieces of information across application windows and thereby guiding the user's attention to relevant information. Applications are coordinated by a management application accessible via a light-weight interface. User selections are synchronized across registered applications and visual links are rendered on top of the desktop content by a window manager. Initial user feedback was very positive and indicates that visual links improve task efficiency when analyzing information from multiple sources.

Keywords: visual links, information analysis, multiple coordinated views

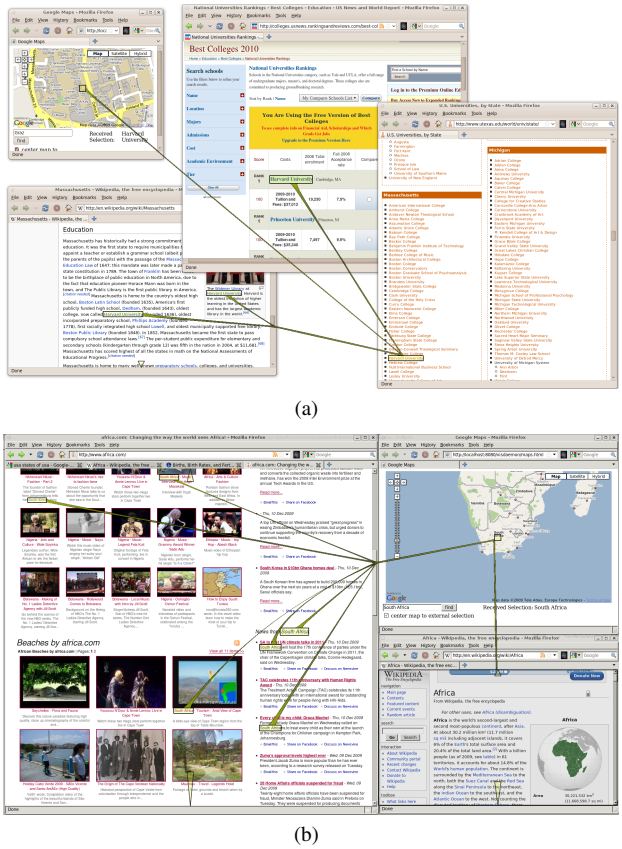
Index Terms: H.5.2 [Information interfaces and presentation (e.g., HCI)]: User Interfaces—Interaction styles (e.g., commands, menus, forms, direct manipulation)

1 Introduction

Information analysis often requires to investigate and evaluate data from different sources in different visual representations. This analysis process may incorporate multiple applications – such as web browsers, document readers, or dedicated visualization software – arranged in multiple windows on a computer desktop. With increasing display size, these application windows can be arranged side-by-side, so all relevant information is simultaneously visible. However, operating systems treat individual application windows independently without coordinating their content, their visual representations, or visualizing correlations across multiple windows. Relating and comparing information across applications therefore requires the user to manually search for relevant items, making the process exhaustive and error-prone.

Multiple coordinated views employ two or more visualization *views* for data investigation (*e.g.*, [14, 7, 25]). The individual views are *linked*, so changes in one view are reflected in all other views.

*e-mail: waldner@icg.tugraz.at
†e-mail: puff@icg.tugraz.at
‡e-mail: lex@icg.tugraz.at
§e-mail: streit@icg.tugraz.at
¶e-mail: schmalstieg@icg.tugraz.at



(a)

(b)

Figure 1: Using visual links to relate information on web pages with geographic locations: Harvard University (a) and South Africa (b).

The application areas for such systems are manifold, for instance: tasks where detailed and contextual information is required at the same time [19], tasks where alternative views on the same data are of interest [6, 21], or tasks where a comparison of different data sets is needed [20]. Multiple coordinated views typically are embedded in one visualization system. However, these visualization systems usually provide only specialized visualization views and data sets for the respective application area. To extend their applicability, unsynchronized separate applications for additional information and views are required in many cases. To overcome this limitation, North and Shneiderman developed a system that coordinates multiple applications through an API [19]. While these approaches help to keep the workspace consistent and provide visual indications of relationships through synchronized highlighting or *brushing* [2], they fail to actively guide the user's attention to relevant items.

When working with multiple application windows on a conventional desktop, a large amount of information is visible – in partic-

ular if the available display space is large [13, 3]. With multiple applications involved, the content is highly heterogeneous, and the user needs to observe different data types – like text, graphs, images, or maps – which may or may not contribute to the primary task. As only a small portion of the display is actively observed by the user [28], most of the information is located in the visual periphery. Guiding the attention to peripheral items or showing relationships between information across application windows thus requires strong visual cues to guide the user's attention to locations of interest and to show relationships and patterns explicitly.

In this paper we present visual links – connection lines linking related data items – across application windows (see Figure 1). Applications communicate with a management application through a light-weight interface. Rendering of the connection lines is accomplished as a plug-in to an OpenGL-based compositing window manager. To demonstrate the usefulness of cross-application visual links we present usage scenarios with three application examples – a web browser, a visualization software, and a map mashup application. Initial user feedback is encouraging and showed that users understood and appreciated the concept of visual links.

2 Related Work

Our work draws from the field of multiple coordinated views, common in information visualization, and builds upon knowledge from display space management research, in particular for large-scale displays.

2.1 Multiple Coordinated Views

Multiple coordinated views have been used in visualization for decades in different application domains (*e.g.*, flow data simulation [7] or biomedical data analysis [25]). While such systems typically embed multiple views in a special-purpose visualization application, Snap-Together Visualization [19] is a notable exception. It provides a light-weight API based on COM and uses a database for data synchronization across views and applications, respectively. Snapped applications provide synchronized mechanisms, such as "load", "select" (synchronized highlighting), and synchronized scrolling. Views are coordinated pairwise based on a set of actions and join relationships. Similar to Snap-Together Visualization, we provide a minimally-invasive interface for cross-application communication of arbitrary applications. However, our work differs as we focus on the effect of *visualizing relationships* across applications, while Snap-Together uses application specific highlighting only.

North and Shneiderman [19] pointed out the strength of coordinated views for tasks involving analysis of overview information and detailed information on demand. Plumlee and Ware [21] showed that multiple windows are more efficient if the number of items to be investigated is high and, as a consequence, the demands on visual memory are high. Baldonado *et al.* [27] set up rules for multiple view usage and recommend to employ them when the information can be represented in diverse ways (like *overview and detail*), if multiple views illustrate correlations, or if complex data can be split into smaller, easily manageable chunks. They also pointed out the importance of making the relationships among multiple views apparent to the user – the key aspect of our work.

Relationships across multiple views are traditionally expressed through highlighting or color coding (for example in *Tableau* [24], *prefuse* [10] or *SimVis* [7]). Only a few multiple view systems use more expressive highlighting mechanisms like line connections. Spiral calendar [18] shows the relationships among multiple calendar hierarchies through half-transparent connections. Shneiderman and Aris [23] link categories of network visualizations to another. To avoid visual clutter, the user can filter the number of links by dynamic queries. In VisLink, Collins and Carpendale [4] visualize the relationships of individual 2D visualization planes arranged in 3D space through multiple edge connections. They argue that these edge connections help to reveal relationships, patterns, and connections between views. Streit *et al.* [25] use visual links to show dependencies between 2D pathways (models of biological processes) and gene activity and developed a 3D view arrangement, the *Bucket*, to show relations between arbitrary views. Bubblesets [5] visualize *set relationships* within existing visualizations. Although this concept can be conceptually applied across multiple views, the implementation is limited to single view applications. Our approach differs from these applications as we do not only link views managed by one application, but include information from different sources and applications, like web pages and maps.

In *LivOlay*, Jiang *et al.* [15] relate two remote desktop views on a large projected display by overlaying the two view planes based on a manual spatial registration. Tan *et al.* [26] replicate arbitrary window regions into independent small *WinCuts*. Their main application areas are convenient side-by-side comparisons of arbitrary data sets and effective spatial organization of information on limited screen space. While WinCuts potentially removes the information items from their contextual surroundings, LivOlay only works for information which is spatially related, like maps. In addition, these approaches both disrupt the user's spatial arrangement of application windows – which is particularly important for the user's task management [16, 3]. In contrast, our approach does not interfere with the user's spatial window layout and combines information from various application sources, thereby dealing with arbitrary data which cannot easily be spatially registered.

2.2 Display Space Management

Relating information from multiple application windows is an increasingly demanding topic, as display sizes, as well as the amount of available information, steadily grow. Investigations of users' display space management on multi-monitor systems [8, 3] and large displays [13, 3] showed that the number of visible application windows increases with the available display size. Hutchings and Stasko [13] showed that users rarely maximize windows on single large displays, but rather "carefully coordinate" multiple windows on the available screen space. The users' main consideration is that they use information from multiple windows to interact with the primary task window. They furthermore observed that users tend to get distracted by irrelevant information shown in secondary windows. Our work focuses on this issue and aims to explicitly visualize information which is relevant to conduct an information analysis task.

Research on perception showed that only a very small portion of the display is in the active user's visual field [28]. Thus, interaction on large or multiple displays requires guidance mechanisms to make users aware of changes in the display and regions of interest. Khan *et al.* [17] use a spotlight metaphor to illuminate the region on a large-scale projection wall, where the presenter is currently interacting. They showed that users more easily spot the cursor on a large display wall using their technique than without any highlighting technique. Rekimoto and Saitoh [22] visualize mouse pointer locations in a multi-display environment by *anchoring* the cursor at a fixed location on the user's laptop and drawing a line connection to the current spot on tabletop or wall display. Hoffmann *et al.* [11] designed visual cues for window switching. They investigated several highlighting techniques for guiding the user's attention to the new active window on a triple monitor setup. They found that combinations of frames and trials are most effective. All those techniques aim to guide the user to a single spot – the location where interaction takes place or new information is appearing. In contrast, we investigate attention guidance to multiple regions of interest scattered arbitrarily on the available display space.

Others have developed techniques to visualize off-screen locations for small-scale displays on handheld devices (*e.g.* [1, 9]). In

contrast, we investigate how *visible* pieces of information can be connected to retrieve their relationships and to see patterns. However, we do consider the aspect of information being available but currently invisible (*e.g.*, when scrolled away in a web browser) and apply an arrow-based off-screen visualization for these situations.

3 Visual Links on the Desktop

Information analysis requires to evaluate, compare, and relate information from different sources to find trends and patterns. To support these tasks, visualization systems use techniques like highlighting or color coding. However, supportive information sources, such as web pages or spreadsheets, lack such functionality. There, information is encoded in structures like text, maps, or images. When working with data sources shown in separate applications, the information is contained in multiple application windows, spatially arranged within the display space, and potentially surrounded by other application windows not contributing to the analysis task. Without coordination of applications and visual linking, the user first has to gain an overview of available information in all application windows, followed by a closer investigation of potentially relevant items in these windows, to finally relate these items in order to identify trends, patterns, and correlations.

Supportive information located at the periphery of the display is likely to be overlooked, even when highlighted. Hoffmann *et al.* [11] found that for targets appearing at a large distance from the focus window, trails to the target location performed better than highlighting the target with a colored frame. They also identified curved, asymmetric trails to be more easily detectable, as they are more distinguishable from the merely rectangular screen content.

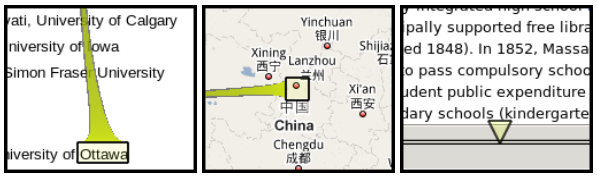
Based on these considerations and inspired by previous work in multiple view visualization (*e.g.*, [4, 25]), we created visual links for the desktop, connecting related, but spatially distributed, data items of various sources and different representations. With visual links we can explicitly and consistently show related items across independent applications.

3.1 Application Coordination

Similar to Snap-Together Visualization [19], our software infrastructure uses a minimally invasive approach to coordinate existing applications. It supports rendering of visual links across applications registered to a management process running in the background. Multiple client applications can register to this process and report selection identifiers upon local user selection. The registered applications themselves determine how a user selection is triggered. Possible selection events are: hovering the mouse pointer over an element, marking text, or entering a search string. The selection ID is sent to the management application as character string where it is distributed to all registered applications. Each client application evaluates the incoming selection ID individually. For instance, in text documents, these selection identifiers correspond directly to a substring of the document (*e.g.*, single words), while in a spreadsheet application, it may be mapped to a whole column of a table with the selection identifier as column headline. To resolve more complex relations, client applications are free to map the incoming selection ID to others. For instance, an application may translate the incoming identifier string so it matches the language of the displayed data set or can determine a suitable hierarchy level for mapping hierarchical data structures. Once a client application has found an entity matching the provided selection ID, it reports the bounding rectangle of the selection, or alternatively, a single point, to the management application, where it is used to render the visual links.

3.2 Visualizing Connections

The *selection regions* reported by the client applications may enclose a string in a text document (Figure 2a), mark a country in a map application (Figure 2b), or highlight an element in a chart. Following Hoffmann *et al.*'s guidelines [11], we indicate related regions by rendering frames around the selection regions, as well as by connection lines from the user's current interaction window to the selection regions. If multiple selection regions are reported in a single application window, connection lines are *bundled* [12] to reduce visual clutter. Bundling also clearly indicates relatedness of selection regions with regard to their application window.



(a) (b) (c)

Figure 2: Close-ups on single connection lines and selection regions highlighting (a) text and (b) a map location. Hidden items on a website are indicated by an arrow (c).

We distinguish two window types: The **source window** is the application window where the user selection has been registered. As the user focus is currently on this window, all visual links to related information emerge from this spot. If multiple selection regions are located in the source window, each region is highlighted and connected to a bundling point in the center of gravity of all regions (*c.f.*, Figure 3 A), where the connections to the target windows emerge.

Target windows are all registered application windows where no user interaction is taking place and at least one selection region was reported. The main purpose of visual links is to lead the user's focus to these windows and to express the relationship of relevant items with respect to the source window. Each target window reporting selection regions is linked with the source window by a single connection line. This connection line bundles the connections to all selection regions in the target window. In contrast to the source window, the bundling point is set to the intersection point of the line connecting source and target window with the target window's boundary. From this bundling point, the individual connection lines to all selection regions emerge (see Figure 3 C-D).

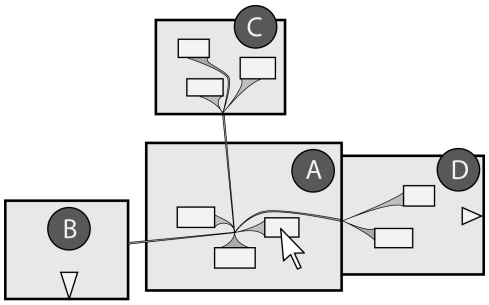


Figure 3: Visual links emerging from a source window (A) to three target windows (B-D). Target windows B and D contain hidden items, indicated by an arrow. Mind the connection lines in window A and C avoiding selection region obstacles.

Connection lines are rendered as Bézier surfaces, emerging from the window's bundling point with a given line width and expanding to the selection region border up to a given maximum width (*c.f.*, Figure 2). These connections are rendered half-transparent, so underlying data can still be identified. "Shadows" surrounding the connections help to discriminate visual links from the desktop content. To avoid occlusion of potentially valuable information, render-

ing connection lines across selection regions is avoided. If a connection line would intersect a selection region, the corner point of the intersected region leading to the shortest non-intersecting path is added to the connection line. This results in the connection being "curved" around the region (Figure 3 A and C).

If an application reports selection regions outside the visible window region, arrows at the window boundaries give a visual cue about invisible related information (see Figure 3 B and D). Contrary to some off-screen visualization techniques encoding the exact location of off-screen items, like *Halos* [1] or *Wedges* [9], we group invisible regions into four off-window directions (up / down / left / right). The width of the arrows is static, while the length encodes the number of hidden selection regions in this direction, giving the user a cue about the amount of invisible information in each direction. For instance, in Figure 1b, the long arrow on the bottom of the left browser window indicates a high number of hidden items, which can be revealed by scrolling the website further down. If a target window has no visible, but only hidden selection regions, visual links are drawn only to the window border (*c.f.*, Figure 3 B).

Connection lines, selection region highlights, and arrows are displayed until a new user selection, a change in window layout, or a change in window content (*e.g.*, by scrolling) is registered. As visual links introduce some visual clutter on the desktop, we provide two methods to alleviate this problem. The user can toggle visual links, to only activate them when needed, by pressing a keyboard shortcut. Additionally, the user can choose to let the links fade out after some time (as shown in Figure 7) and can fade them back in whenever they are useful again.

4 Implementation

Our visual links software infrastructure is divided into three layers (see Figure 4). The central coordinating instance between the client applications and the rendering layer is the *Visual Links Manager*. It collects the selection regions of the client applications and communicates with the rendering layer via a remote procedure call (RPC) interface. The *Visual Links Renderer* is the layer responsible for routing and rendering of the visual links. Client applications can utilize the interface exposed by the Visual Links Manager by implementing it either directly or through an add-on.

In this section we discuss those three components, followed by a description of three sample applications in Section 5.

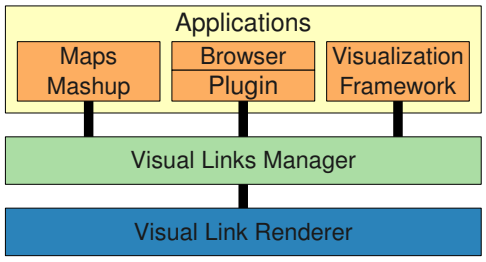


Figure 4: System architecture: client applications connect to the Visual Links Manager, which coordinates selections and forwards selection regions to the Visual Links Renderer, which routes and renders the visual links.

4.1 Visual Links Manager

The Visual Links Manager is responsible for propagating selection identifiers between registered applications and to collect the associated bounding rectangles for all selection regions. It is based on the Tomcat Java Application Server[1]. The servlet container of Tomcat provides an HTTP-based connection layer which is capable of handling multiple concurrent connections. This way, web browser applications or plug-ins can directly communicate with the Visual Links Manager without an additional communication layer.

The Visual Links Manager receives the selection ID and associated selection regions from the source window, *i.e.*, the window where the user is currently interacting. The selection ID is then reported to all remaining registered applications (target windows), which send the selection regions associated with the selection ID as a response. After all registered applications have reported their selection regions, the Visual Links Manager delivers the list of regions to the Visual Links Renderer.

4.2 Visual Links Renderer

Visual links are rendered by a plug-in for the OpenGL compositing window manager *Compiz*[2], which is now the default window manager for popular Linux distributions like Ubuntu. The visual links plug-in launches a C++ application which provides an *Ice*[3] RPC interface for the Visual Links Manager application. As input, this renderer application receives a list of selection regions associated with application windows. It then calculates the connection lines as well as the size and position of the arrows that indicate hidden selections. The resulting graphics primitives are sent to the window manager plug-in via a Compiz-internal communication routine. The plug-in renders the primitives on top of all desktop application windows and triggers a desktop repaint whenever links are received or need to be animated for fade-out. Bézier connection lines are rendered as two-dimensional OpenGL evaluators.

Using a compositing window manager allows us to render visually appealing links on top of arbitrary application windows in real-time. However, usage of the Compiz window manager limits us to the X Window System. To make our system available to a wider user base, we also developed an alternative Java-based rendering routine, which is universally deployable, but results in reduced visual quality of the links.

4.3 Application Integration

An application utilizing the visual links software API needs to support three basic actions. First, the application has to *register* with the Visual Links Manager – either automatically at start-up or by user request – and report the visible window region to the Visual Links Manager. Second, it has to provide a user interface to *trigger source selections*, find other matches for the current local selection, and deliver both, a selection ID and a bounding rectangle for every selection, to the Visual Links Manager. Third, it needs to be able to *process incoming selections* from the Visual Links Manager when acting as a target window. Similarly to source selections, it has to find matching entities and send the bounding rectangles back to the Visual Links Manager. Additionally, a client application needs to be aware of changes in the window content (*e.g.*, if the user scrolls the content) and re-trigger source selections to update the selection regions. Application support can be grouped into three categories:

Direct support: A software can be extended to directly utilize the interface provided by the Visual Links Manager.

Mashup: A web mashup combines functionality from an existing API with the Visual Links Manager interface.

Software extensions: If an existing software supports extension mechanisms (*e.g.*, via plug-ins), the required functionality can be implemented in a minimally invasive manner.

The last category allows for usage of a wide range of applications at minimal effort, as many common web browsers and office applications provide extension interfaces. With these applications, a variety of use cases can be covered, because information from the web and common document formats are easily supported. However, when using such applications, data access is often restricted

[1] http://tomcat.apache.org

[2] http://www.compiz.org

[3] http://www.zeroc.com

to textual content. As a consequence, ID mapping is limited to text parsing and string comparisons. In contrast, applications such as visualization frameworks often provide advanced interaction techniques for item selections and have ID mapping systems available – but often lack the extendibility required for a minimally invasive integration.

5 Application Examples

As sample applications we chose to use one application which can be extended via an add-on, one mashup application, and one application directly implementing the interface. For the first, we created an add-on for a popular web browser to include information from the web. For the second category, we realized a map mashup application utilizing a powerful maps API for visualizing and linking geographic locations. For the last, we extended an existing visualization framework.

5.1 Web Browser Add-On

We implemented an add-on for the popular cross-platform web browser *Mozilla Firefox*[4]. The add-on has full access to the DOM (document object model) of the displayed HTML-document. This feature is utilized to find occurrences of the given selection ID string within the text passages of the document. By temporarily enclosing the matching strings with an HTML-`<span>` tag, the position and size of the selection regions within the browser window can be retrieved. Communication to the Visual Links Manager to exchange selection IDs and selection region information is based on the `XMLHttpRequest` feature supported by the Firefox web browser.

The user can choose for every browser window whether it should be connected to the Visual Links Manager and thereby retains full control over which web browser windows contribute to the information analysis task. User selections are triggered by selecting text on the displayed website and pressing a button embedded in the browser UI.

5.2 Map Mashup Application

Our map mashup application consists of a single HTML-page, utilizing JavaScript and the *Google Maps API*[5] for an interactive map application resembling standard Google Maps, but with added visual links. The application registers to the Visual Links Manager whenever the page is loaded. Similar to the Firefox add-on, the map application utilizes the `XMLHttpRequest` feature of Firefox for communication. When receiving a selection ID string, the Google Maps API is queried for an associated geographic location. The location obtained from the first search hit is then converted to screen coordinates and a small bounding rectangle around this position is reported to the Visual Links Manager as selection region. The user can choose whether the map should be centered and zoomed to the retrieved geographic location or if the map should remain static – showing, for example, an overview of the world where visual links point to the selected region.

To trigger a geographical search and to show visual links emerging from the map application, the user can enter a location search string. This search string is then reported to the Visual Links Manager as selection ID together with the resulting selection region.

5.3 Visualization Framework

We extended *Caleydo* [25], an existing multiple coordinated view visualization framework which provides common visualization techniques, like a heat map, parallel coordinates and a radial layout, by directly implementing the Visual Links Manager interface. While Caleydo originates from the biomedical domain it is able to handle arbitrary data. The framework supports linking and brushing across all views as well as diverse filter operations in the views.

In addition, the framework uses an internal 3D view management of 2D views and connects related items within the application with visual links. However, these visual links are restricted to this specialized compound 3D view – valuable meta-information from other sources can not be integrated.

Whenever elements are selected, they are highlighted by the framework itself, and the connection coordinates, together with the selection ID, are sent to the Visual Links Manager.

Caleydo uses a sophisticated ID mapping system to resolve ID relations within the biomedical domain. There is a wide range of common gene identifiers, where n-to-m mappings are possible. We utilize this ID management for incoming as well as for outgoing IDs, so that Caleydo understands many different identifiers.

6 Usage Scenarios

To illustrate the applicability of our visual links solution, we present three usage scenarios.

6.1 Demographic and Economic Statistics Analysis

Comparative studies of demographic and economic statistics require analysis of multivariate data sets. The geographic locations of the countries to be compared form the basic layer implicitly given by the data. While a geospatial representation can only encode a limited number of attributes (*e.g.*, by color-coding), visualization techniques for multi-dimensional data, such as parallel coordinates, can lead to loss of geospatial context information.

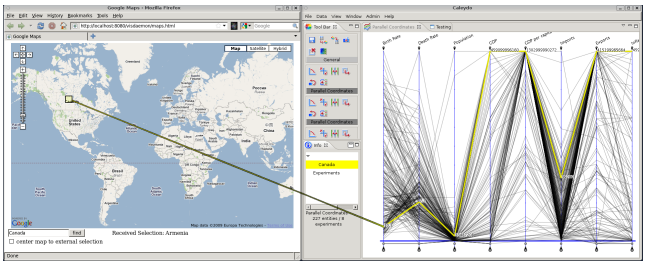


Figure 5: Comparing economic statistics in parallel coordinates and visualizing the geospatial context in the linked map application.

In this scenario, we registered the modified visualization framework and our map application to the Visual Links Manager and loaded the statistics data into the parallel coordinates view. By hovering over the polylines in the parallel coordinates, the search string for the line – the country's name – is propagated to the Visual Links Manager and forwarded to the map application. Visual links show the connection from the selected polyline in the parallel coordinates to the location in the map. Figure 5 shows economic data for Canada where the birth rate and death rate are low, the population is rather small, but GDP and exports are among the highest of all countries. This way, the full amount (or a filtered sub-set) of available data can be explored and conveniently compared with a visualization system, while the geospatial context is provided by the map application.

6.2 Biomedical Analysis

Information about the function and the dependencies of genes are available in abundance in diverse online databases, like Entrez Gene[6] or KEGG[7]. When an expert explores a condition, he often reads information contained in these databases.

[4] http://www.mozilla.com/firefox/
[5] http://code.google.com/apis/maps/
[6] http://www.ncbi.nlm.nih.gov/gene/
[7] http://www.genome.jp/kegg/

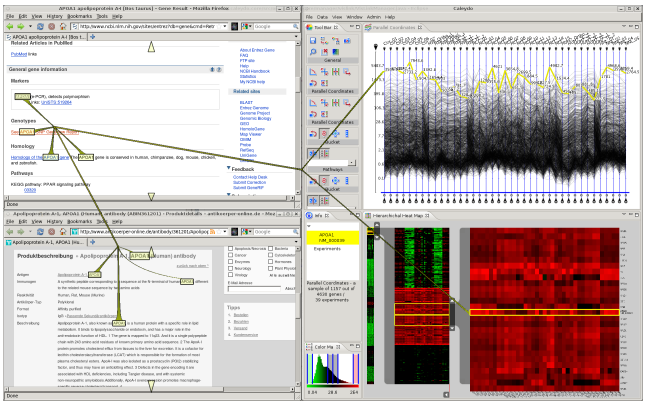


Figure 6: Comparing gene expressions from an experiment in a parallel coordinates and a heat map view with two gene databases on the web.

Measurements of the gene activity acquired with high throughput methods enable life scientists to get snapshots of the activity on a whole-genome scale at low cost. As a result, lots of data is generated, which is analyzed using statistics and visualization. Methods such as parallel coordinates or heat maps are useful when filtering or exploring concrete gene activity values.

By visually relating the information found in the literature to concrete values in an information visualization system, as shown in Figure 6, expert users can easily identify the activity levels of relevant genes across all experiments and therefore save time and effort trying to manually find matches between concrete datasets and information in the literature.

6.3 Document Organization

In daily work and spare time people are often confronted with search tasks, where information from multiple sources needs to be related. Consider an online shopping scenario, as illustrated in Figure 7: a user reads hardware specifications in an online magazine and browses online stores at the same time. The user can trigger visual links from the magazine to look up if the online store sells the item of interest. On the other hand, the user can browse the online store for inexpensive items and quickly check the magazine website for comments and test results.

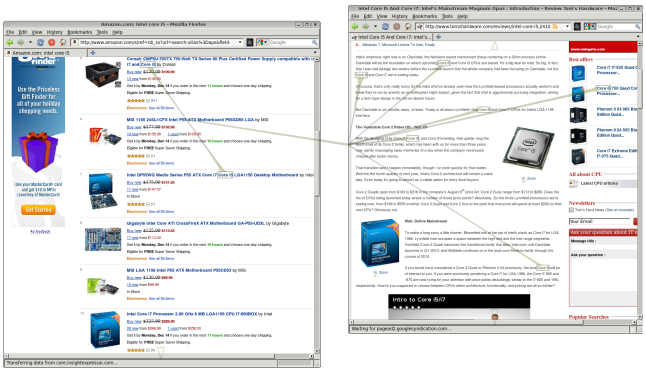


Figure 7: Comparing hardware prices and specifications. Visual links are faded to a low alpha value to reduce visual clutter.

A similar workflow is required in the following more complex scenario concerning the guest editor of a journal. After the submission deadline, around 20 submissions will be sent to the guest editor for which he needs to find at least three reviewers. Finding suitable reviewers from the field without conflicts of interest, necessitates the guest editor to switch forth and back between a PDF viewer showing the submitted article, an e-mail application to check communication with the editor-in-chief of the journal, the journal's reviewing website, and a web browser to find out current contact details of potential reviewers. The data to be related is linked by the submission number of the article or a potential reviewer's name. For example, one typical interaction is to look at the list of references for a specific paper and identify cited authors who would make good reviewers. With the help of visual links, the guest editor can quickly check whether the name has been discussed in an e-mail with the editor-in-chief or is contained in the authors list of the submitted paper on the journal's reviewing website, simply by selecting a potential reviewer name from the paper's reference list.

This example has not been realized in practice, because at the time of writing not all described applications have been instrumented with visual links. However, both popular PDF reader and e-mail programs have plug-in interfaces and are therefore easily extensible.

7 User Feedback

To evaluate user acceptance and usability of the visual links concept, we conducted an informal user evaluation with seven participants (aged 25 to 39). Participants were recruited from a local university and a software development company.

7.1 Task

Users were asked to accomplish an information analysis task utilizing information from multiple sources. We arranged four application windows on a 26" monitor with 1920x1200 pixels, as shown in Figure 8. On the left, the map application was placed showing a static view of the African continent. On the right, we placed the visualization framework showing demographic and economic statistics of 190 countries in a parallel coordinates view. In between, an HTML-page in a web browser listed all African countries exporting oil, while a second web browser window below listed all African countries. Prior to the actual task, users were trained on parallel coordinates and how to invoke visual links from the respective applications.

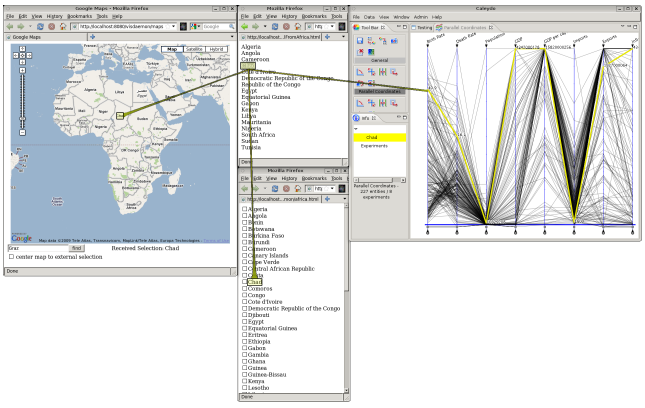


Figure 8: Window arrangement for the user evaluation: the map application fixed to Africa, two websites with a list of countries, and a parallel coordinates view of demographic and economic statistics of 190 countries. Visual links visualize correspondences between the individual windows.

The task was to find all African countries north of the equator, which export oil and have a birth rate (*i.e.*, number of childbirths per 1000 people per year) higher than 30. The birth rate data was plotted on the first axis in the parallel coordinates view. Countries

fulfilling these requirements had to be marked on the HTML-page containing the list of all African countries. We chose the example of Africa, as demographic and economic statistics were strongly diverse across the countries and we expected our users to be relatively unfamiliar with the exact geographic locations of African countries. We measured the task completion time, recorded observations, and collected user comments in an interview.

7.2 Observations

All but two users could successfully complete the task with no errors. They required 1:40 to 4:30 minutes to complete the task.

One user did not succeed as she missed the small country Equatorial Guinea, which is actually located north of the equator but hard to identify on the map with the applied zoom level. Two other users had to zoom the map to determine the exact location of the country. One user mentioned that selection regions are not optimally placed in the map application. As the Google Maps API only delivers the geographical location of the center of the country and corresponding screen coordinates, we do not have control over label placement and do not have the possibilities to highlight the actual country outline. Instead, we use a small bounding rectangle of fixed size to visualize the geographic location. As a consequence, the size of the region does not indicate the size of the country. Additionally, connection lines often cover the label of the country which is – despite transparencies – hard to read.

One user quit the task after four minutes without finding any country fulfilling the requirements. Contrary to all other users, she mainly employed the parallel coordinates view to invoke visual links. However, as the parallel coordinates contained data from 190 countries, it was tedious to find the polylines for the required African countries. All other participants used the browser window with the list of oil-exporting African countries as source window for visual links, as the information was already strongly filtered with only 17 potential countries. From this window, they simply had to follow the link to the parallel coordinates and check the birth rate and the link to the map to see the geographical location. Finally, they followed the link to the browser window containing the list of all African countries to mark the result. The fastest participant could correctly solve the task in one minute and 40 seconds using this strategy.

We generally observed difficulties when users employed the parallel coordinates view as source window. Apart from the one user not finishing the task, we noticed four other users moving the mouse pointer to the parallel coordinates after selecting the country in the browser window. As selection was triggered by mouse-over in the parallel coordinates, their previous selection was lost when the mouse pointer was accidentally moved across a different polyline and they had difficulties finding the polyline for the previously selected country again.

7.3 Feedback and Discussion

User feedback was consistently positive, and was also helpful in terms of identifying user interface problems. Users described working with visual links as “It just works and it is efficient” and “It is fun”. All users agreed that having coordinated windows with visual links is helpful for such a task. They stated that it was clear what visual links visualize and how items were related. Distraction caused by the high amount of visible information was reduced as “it shows only the most essential information”. One user noted that “otherwise [without visual links], the task would take ten times longer”. Except for the one user not being able to find any countries, all users agreed that task completion was very efficient using visual links. One user explained he was fast as he would simply trust the visual links and would not check whether the correct items were connected. Another user said he always double-checked the connections. Surprisingly, this participant completed the task fastest.

However, some users doubted whether visual links are useful for every-day work (for software engineers). Of course, visual links are not necessarily suitable for every task and not indented for constant usage. Instead, we envision visual links to be an on-demand feature which can easily be switched on for information analysis and comparison tasks. The integration of visual links rendering in the window manager and the invocation through keyboard shortcuts allows for such a temporary usage.

Several users commented on the interaction techniques provided by the respective applications. As we could also observe during the task, accurate selection in the parallel coordinates view was tedious – in particular with this high amount of information available. Invoking visual links by moving the mouse pointer across a densely populated parallel coordinates view helps to get a quick impression of the data but is not suitable when accurate selections are required. We therefore now use visual links with a more explicit selection mechanism (mouse click), to let the user more easily control the selected items.

Two users also noted that selecting text in the browser window is complicated, especially if multiple words need to be selected. We therefore have to think of easier methods to select multiple words and will integrate a keyboard shortcut as alternative to the button for invoking visual links. One user suggested to employ an “auto-invocation”, which automatically triggers visual links shortly after a word has been selected.

Another user demanded multi-selections, so he could select multiple countries in the browser and see the respective connections to the visualization software and the map application. With multi-selection, users could have selected all oil-exporting African countries in the first browser window, where each of these selections would have received their own visual links. This feature would reduce the required user intervention, as invoking visual links is required only once. However, it is subject to further research whether the additional line connections introduce so much visual clutter that extracting related information becomes more tedious.

8 Conclusion and Future Work

We created a software infrastructure to visually link related items across applications. This software infrastructure can be utilized by either directly supporting the provided interface or by extending existing applications through add-ons or mashups. We presented application examples for all three categories and showed several usage scenarios which can be accomplished with these applications. The presented system provides an easy way to extend existing applications, such as visualization software, by additional, contextual information. Thereby, the applicability of the application is enlarged, while the required modifications are minimally invasive.

Informal user observation and feedback encourages our approach of visual links across applications. Users could solve a task of combining information from multiple sources very quickly, they clearly understood the concept of visual links, and could easily identify relations between multiple sources. We can therefore summarize the benefits of visual links for information seeking tasks:

- Visual links guide the user’s attention to secondary, peripheral windows and specific items contained therein which are relevant for the primary task.
- Visual links make relevant information in secondary windows more distinct, making it easier for the user to neglect a large amount of information and to concentrate on the primary task.
- Visual links prevent the user from having to search information manually for each involved application, thereby limiting the error probability induced by overseeing information and the effort for the user.

Users also raised some ideas for improvements, such as the ability to make multiple selections, providing easier interaction techniques for selecting words in the web browser, or making selections in parallel coordinates more explicit, so selections are not lost when moving the mouse pointer over the view. We will address these user comments in future work.

The main focus of this work was to express the relationship of *visible* information. However, when working with smaller displays or more application windows, the issue of hidden content has to be considered. Our visual links renderer does consider hidden content, but only if the selection region lies outside the visible window rectangle. In the future, we additionally want to consider content obscured by overlapping windows.

In addition, we aim to extend the presented technique to a distributed system to construct projected tiled displays or multi-display environments (like, for instance, [22]) operated by a PC cluster. As all components communicate via remote interfaces, the system can easily be extended for such scenarios. With a higher number of pixels and multiple displays arranged arbitrarily in the environment, the issue of seeking and relating information will become even more challenging.

Acknowledgements

This work was supported by the Austrian Research Promotion Agency FFG (BRIDGE 822716 and 385567) and the Austrian Science Fond FWF (L427-N15).

References

[1] P. Baudisch and R. Rosenholtz. Halo: a technique for visualizing off-screen objects. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 481–488. ACM, 2003.

[2] R. A. Becker and W. S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.

[3] X. Bi and R. Balakrishnan. Comparing usage of a large high-resolution display to single or dual desktop displays for daily work. In *CHI '09: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1005–1014. ACM, 2009.

[4] C. Collins and S. Carpendale. Vislink: Revealing relationships amongst visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1192–1199, 2007.

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

[6] G. Convertino, J. Chen, B. Yost, Y.-S. Ryu, and C. North. Exploring context switching and cognition in dual-view coordinated visualizations. In *CMV '03: Proceedings of the conference on Coordinated and Multiple Views In Exploratory Visualization*, page 55. IEEE Computer Society, 2003.

[7] H. Doleisch. Simvis: interactive visual analysis of large and time-dependent 3d simulation data. In *Winter Simulation Conference*, pages 712–720, 2007.

[8] J. Grudin. Partitioning digital worlds: focal and peripheral awareness in multiple monitor use. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 458–465. ACM, 2001.

[9] S. Gustafson, P. Baudisch, C. Gutwin, and P. Irani. Wedge: clutter-free visualization of off-screen locations. In *CHI '08: Proceeding of the SIGCHI conference on Human factors in computing systems*, pages 787–796. ACM, 2008.

[10] J. Heer, S. K. Card, and J. A. Landay. prefuse: a toolkit for interactive information visualization. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 421–430. ACM, 2005.

[11] R. Hoffmann, P. Baudisch, and D. S. Weld. Evaluating visual cues for window switching on large screens. In *CHI '08: Proceeding of the SIGCHI conference on Human factors in computing systems*, pages 929–938. ACM, 2008.

[12] D. Holten and J. J. van Wijk. Force-directed edge bundling for graph visualization. In *11th Eurographics/IEEE-VGTC Symposium on Visualization (Computer Graphics Forum; Proceedings of EuroVis 2009)*, pages 983 – 990. IEEE Computer Society, 2009.

[13] D. R. Hutchings and J. Stasko. Revisiting display space management: understanding current practice to inform next-generation design. In *GI '04: Proceedings of Graphics Interface*, pages 127–134. Canadian Human-Computer Communications Society, 2004.

[14] A. S. Jacobson, A. L. Berkin, and M. N. Orton. Linkwinds: interactive scientific data analysis and visualization. *Commun. ACM*, 37(4):42–52, 1994.

[15] H. Jiang, D. Wigdor, C. Forlines, M. Borkin, J. Kauffmann, and C. Shen. Livolay: interactive ad-hoc registration and overlapping of applications for collaborative visual exploration. In *CHI '08: Proceeding of the SIGCHI conference on Human factors in computing systems*, pages 1357–1360. ACM, 2008.

[16] E. Kandogan and B. Shneiderman. Elastic windows: evaluation of multi-window operations. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 250–257. ACM, 1997.

[17] A. Khan, J. Matejka, G. Fitzmaurice, and G. Kurtenbach. Spotlight: directing users' attention on large displays. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 791–798. ACM, 2005.

[18] J. D. Mackinlay, G. G. Robertson, and R. DeLine. Developing calendar visualizers for the information visualizer. In *UIST '94: Proceedings of the ACM symposium on User interface software and technology*, pages 109–118. ACM, 1994.

[19] C. North and B. Shneiderman. Snap-together visualization: a user interface for coordinating visualizations via relational schemata. In *AVI '00: Proceedings of the working conference on Advanced visual interfaces*, pages 128–135. ACM, 2000.

[20] H. Piringer, W. Berger, and H. Hauser. Quantifying and comparing features in high-dimensional datasets. In *IV*, pages 240–245, 2008.

[21] M. D. Plumlee and C. Ware. Zooming versus multiple window interfaces: Cognitive costs of visual comparisons. *ACM Trans. Comput.-Hum. Interact.*, 13(2):179–209, 2006.

[22] J. Rekimoto and M. Saitoh. Augmented Surfaces: A Spatially Continuous Workspace for Hybrid Computing Environments. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 378–385. ACM, 1999.

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

[24] C. Stolte, D. Tang, and P. Hanrahan. Polaris: a system for query, analysis, and visualization of multidimensional databases. *Commun. ACM*, 51(11):75–84, 2008.

[25] M. Streit, A. Lex, M. Kalkusch, K. Zatloukal, and D. Schmalstieg. Caleydo: connecting pathways and gene expression. *Bioinformatics*, 25(20):2760–2761, October 2009.

[26] D. S. Tan, B. Meyers, and M. Czerwinski. Wincuts: manipulating arbitrary window regions for more effective use of screen space. In *CHI '04: extended abstracts on Human factors in computing systems*, pages 1525–1528. ACM, 2004.

[27] M. Q. Wang Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for using multiple views in information visualization. In *AVI '00: Proceedings of the working conference on Advanced visual interfaces*, pages 110–119. ACM, 2000.

[28] C. Ware. *Information visualization: perception for design*. Morgan Kaufmann, 2000.