Data Visualization (DSC 530/CIS 602-01)

Focus+Context

Dr. David Koop
Restaurant locations are derived from the New York City Department of Health and Mental Hygiene database. Due to the limitations of the Health Department’s database, some restaurants could not be placed.
Filtering Attributes

[D. Koop, DSC 530, Spring 2018]

[Yang et al., 2003]
Aggregation: Histograms

• Very similar to bar charts
• Often shown without space between (continuity)
• Choice of number of bins
  - Important!
  - Viewers may infer different trends based on the layout

[Muñzner (ill. Maguire), 2014]
Aggregation in 2D

- Hexagonal bins are more circular
- Distance to the edge is not as variable
- More efficient aggregation around the center of the bin
Modifiable Areal Unit Problem

In cartography, changing the boundaries of the regions used to analyze data can yield dramatically different results.
Modifiable Areal Unit Problem

In cartography, changing the boundaries of the regions used to analyze data can yield dramatically different results.
Modifiable Areal Unit Problem

In cartography, changing the boundaries of the regions used to analyze data can yield dramatically different results.
Test 2

• Details

• This Wednesday, April 11

• Similar Format

• Covers material since the beginning of class but with an emphasis on the material covered since Test 1
Assignment 4 & Project

- Keep working on your project
- Refine designs, implement prototypes
- Assignment 4 will focus on interaction and multiple views
- Out soon
Aggregation: Four Distributions, One Boxplot

Normal

Bimodal

Peaked

Skewed

Box plot

[C. Choonpradub and D. McNeil, 2005]
Attribute Aggregation

- Remember reducing attributes—use statistics: either one variable matches another or doesn't change!
- We can also use similar criteria for aggregating attributes
- **Cluster** similar attributes together
  - How?
K-Means

Run

[C. Polis, 2014]
K-Means Issues

Shape

Number of Clusters

[D. Robinson, 2015]
Dimensionality Reduction

• Attribute Aggregation: Use fewer attributes (dimensions) to represent items

• Combine attributes in a way that is more instructive than examining each individual attribute

• Example: Understanding the language in a collection of books
  - Count the occurrence of each non-common word in each book
  - Huge set of features (attributes), want to represent each with an aggregate feature (e.g. high use of "cowboy", lower use of "city") that allows clustering (e.g. "western")
  - Don't want to have to manually determine such rules

• Techniques: Principle Component Analysis, Multidimensional Scaling family of techniques
Principle Component Analysis (PCA)

Original data space vs. component space.

Gene 1, Gene 2, Gene 3.

PC 1, PC 2.

[Source: M. Scholz, CC-BY-SA 2.0]
Principal component analysis (PCA) is a technique used to emphasize variation and bring out strong patterns in a dataset. It's often used to make data easy to explore and visualize.

First, consider a dataset in only two dimensions, like (height, weight). This dataset can be plotted as points in a plane. But if we want to tease out variation, PCA finds a new coordinate system in which every point has a new (x,y) value. The axes don't actually mean anything physical; they're combinations of height and weight called “principal components” that are chosen to give one axes lots of variation.

Drag the points around in the following visualization to see PC coordinate system adjusts.

[Principal Component Analysis Explained, Explained Visually, V. Powell & L. Lehe, 2015]
Here's the plot of the data along the first principal component. Already we can see something is different about Northern Ireland.

Now, see the first and second principal components, we see Northern Ireland a major outlier. Once we go back and look at the data in the table, this makes sense: the Northern Irish eat way more grams of fresh potatoes and way fewer of fresh fruits, cheese, fish and alcoholic drinks. It's a good sign that structure we've visualized reflects a big fact of real-world geography: Northern Ireland is the only of the four countries not on the island of Great Britain. (If you're confused about the differences among England, the UK and Great Britain, see: this video.)

For more explanations, visit the Explained Visually project homepage.
Non-linear Dimensionality Reduction

\[ \Phi_{gen} : \mathcal{Z} \rightarrow \mathcal{X} \]

\[ \Phi_{extr} : \mathcal{X} \rightarrow \mathcal{Z} \]

original data space $\mathcal{X}$

component space $\mathcal{Z}$

[M. Scholz, CC-BY-SA 2.0]
Dimensionality Reduction in Visualization

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Nodes</th>
<th>Dimensions</th>
<th>Glimmer</th>
<th>Hybrid</th>
<th>PivotMDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>cancer</td>
<td>683</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>shuttle_big</td>
<td>43,500</td>
<td>9</td>
<td>0.22 s</td>
<td>1.99 s</td>
<td></td>
</tr>
<tr>
<td>grid</td>
<td>10,000</td>
<td>8</td>
<td>0.59 s</td>
<td></td>
<td>1.67e-4</td>
</tr>
<tr>
<td>docs</td>
<td>28,433</td>
<td>28,374</td>
<td>2.11 s</td>
<td></td>
<td>0.157</td>
</tr>
</tbody>
</table>

Fig. 8. MDS layouts showing visual quality, time, and stress for the Glimmer, Hybrid, and PivotMDS algorithms. The data set name, the number of nodes (N), and the number of dimensions (D) appear above each column. Time in seconds appears at the bottom left of each entry, with normalized stress on the bottom right.

[Glimmer, Ingram et al., 2009]
Tasks in Understanding High-Dim. Data

Task 1
- Input: High-dimensional data
- Output: 2D data

What?
- In High-dimensional data
- Out 2D data

Why?
- Produce
- Derive

Task 2
- Input: 2D data
- Output: Scatterplot Clusters & points

What?
- In 2D data
- Out Scatterplot
- Out Clusters & points

Why?
- Discover
- Explore
- Identify

How?
- Encode
- Navigate
- Select

Task 3
- Input: Scatterplot Clusters & points
- Output: Labels for clusters

What?
- In Scatterplot Clusters & points
- Out Labels for clusters

Why?
- Produce
- Annotate

[Munzner (ill. Maguire), 2014]
Probing Projections

Julian Stahnke, Marian Dörk, Boris Müller, and Andreas Thom

Abstract
—We introduce a set of integrated interaction techniques to interpret and interrogate dimensionality-reduced data. Projection techniques generally aim to make a high-dimensional information space visible in form of a planar layout. However, the meaning of the resulting data projections can be hard to grasp. It is seldom clear why elements are placed far apart or close together and the inevitable approximation errors of any projection technique are not exposed to the viewer. Previous research on dimensionality reduction focuses on the efficient generation of data projections, interactive customisation of the model, and comparison of different projection techniques. There has been only little research on how the visualization resulting from data projection is interacted with. We contribute the concept of probing as an integrated approach to interpreting the meaning and quality of visualizations and propose a set of interactive methods to examine dimensionality-reduced data as well as the projection itself. The methods let viewers see approximation errors, question the positioning of elements, compare them to each other, and visualize the influence of data dimensions on the projection space. We created a web-based system implementing these methods, and report on findings from an evaluation with data analysts using the prototype to examine multidimensional datasets.

Index Terms
—Information visualization, interactivity, dimensionality reduction, multidimensional scaling.

1INTRODUCTION
A primary goal of information visualization is to find patterns and relationships in multivariate datasets. Many visualization techniques have been developed towards this goal such as multiple coordinated views [2], parallel coordinates [14], scatterplot matrices [28], and dimensionality reductions such as multidimensional scaling (MDS) and principal component analysis (PCA) [5]. Dimensionality reductions are a particular class of techniques that synthesise high-dimensional data spaces onto projection spaces with much fewer dimensions, typically the two dimensions of the plane. While most visualization techniques juxtapose the different data dimensions as matrices or columns, dimensionality reductions integrate them into a planar canvas. The projection results in a so-called spatialisation (i.e., embedding) of data elements that approximately represents similarity as proximity and in turn dissimilarity as distance. Considering that the human perceptional system comprises a well-developed capacity for spatial reasoning, the assumption is that spatialisation would be a more natural way [31] to analyse high-dimensional datasets since groupings, separations, and other patterns among data elements become immediately discernible.

However, there are two major caveats linked with dimensionality reduction: first, it can be challenging to interpret the positions of projected elements, and second, the errors that occur with any projection technique are not exposed to the viewer. Previous research on dimensionality reduction focuses on the efficient generation of data projections, interactive customisation of the model, and comparison of different projection techniques. We contribute the concept of probing as an integrated approach to interpreting the meaning and quality of visualizations and propose a set of interactive methods to examine dimensionality-reduced data as well as the projection itself. The methods let viewers see approximation errors, question the positioning of elements, compare them to each other, and visualize the influence of data dimensions on the projection space. We created a web-based system implementing these methods, and report on findings from an evaluation with data analysts using the prototype to examine multidimensional datasets.

D. Koop, DSC 530, Spring 2018
Probing Projection Goals

- Examining the Projection
- Exploring the Data
- Design Goals:
  - Show and correct approximation errors
  - Allow for multi-level comparisons
  - Spatial orientation
  - Consistent design
- Allow **grouping** of samples
  - Selections
  - Classes
  - Clusters

[J. Stahnke et al., 2015]
Tooltips with statistics

Portugal

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Value</th>
<th>Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Educational attainment</td>
<td>35</td>
<td>-2.4 σ</td>
</tr>
<tr>
<td>Employees working</td>
<td>9.31</td>
<td>-0.034 σ</td>
</tr>
<tr>
<td>Life expectancy</td>
<td>80.8</td>
<td>+0.39 σ</td>
</tr>
<tr>
<td>Life satisfaction</td>
<td>5.2</td>
<td>-1.6 σ</td>
</tr>
<tr>
<td>Self-reported health</td>
<td>50</td>
<td>-1.3 σ</td>
</tr>
<tr>
<td>Student skills</td>
<td>488</td>
<td>-0.20 σ</td>
</tr>
<tr>
<td>Time devoted to leisure</td>
<td>14.95</td>
<td>+0.13 σ</td>
</tr>
<tr>
<td>Years in education</td>
<td>17.8</td>
<td>+0.31 σ</td>
</tr>
</tbody>
</table>

correct distances

[J. Stahnke et al., 2015]
Comparing Two Groups

South America 3 samples
Northern Europe 9 samples

<table>
<thead>
<tr>
<th>Dimension</th>
<th>South America</th>
<th>Northern Europe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Educational attainment</td>
<td>50</td>
<td>77</td>
</tr>
<tr>
<td>Employees working vs. full-time</td>
<td>18</td>
<td>6.2</td>
</tr>
<tr>
<td>Life expectancy</td>
<td>75</td>
<td>81</td>
</tr>
<tr>
<td>Life satisfaction</td>
<td>7.1</td>
<td>7.4</td>
</tr>
<tr>
<td>Self-reported health</td>
<td>65</td>
<td>77</td>
</tr>
<tr>
<td>Student skills</td>
<td>420</td>
<td>500</td>
</tr>
<tr>
<td>Time devoted to leisure</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>Years in education</td>
<td>16</td>
<td>19</td>
</tr>
</tbody>
</table>

[D. Koop, DSC 530, Spring 2018]
Heatmap from Dimension Hover

4.4 Analysing dimensions

...it is important to be able to quickly reference original dimensions...
Showing Error via Sample-centric Halos

Fig. 5. Comparing an individual sample with a class from the well-known Iris flower data set. In addition to the distribution of dimensions of the class and the value of the sample, the visualization also indicates sample-centric distortions using grey and white halos.

4.4.2 Density plots

While the heatmaps show how the values are spatially distributed in the projection space, kernel density plots in the list of dimensions show their value distributions. In the prototype, currently the plots are generated roughly equivalent to R's bw.nrd0 function which uses Silverman's 'rule of thumb' [24, p. 48]. Percentiles are indicated on the density plots to support the visual assessment. Used together with brushing and linking, it is possible to assess how a sample, or a group of samples, relates to the whole dataset.

Markers or sub-plots for selected elements are shown on the density plots in the list of dimensions, providing dynamic highlights of samples being examined (see Figure 5, lower right). Additional markers display the dimensional values mouse position in the projection space, based on the calculations done for the heatmap, making it possible to gradually track value progressions for multiple dimensions.

4.5 Examining projection errors

Besides exploring the distribution of samples and dimensions, the visualization environment allows for the integrated examination of projection errors by providing per-sample halos, distance corrections, and a dendrogram.

4.5.1 Error halos

The cumulative distance error for each point is displayed as a halo around the dot, with the radius corresponding to the relative amount of error and the value indicating the direction of the error (see Figure 6). This is intended to help the user visually understand the quality of the projection and find potentially unreliable spots. Hovering over a dot shows the errors in relation to the hovered point, to check if the distances between certain points are correct or just projection artefacts, and learn which points should be closer together or further apart.

Halos were chosen because their visual properties are a good match for the properties of the error they represent. The error does not belong to the sample, but to the projection, and as such should be displayed by the projection. A halo is clearly connected to the dot, but also part of the projection. Size was chosen as a very intuitive metric to display the amount of error, with points with a large error standing out easily. The brightness of the halo displays the direction of the error. If the other points are farther away in the projection than in the high-dimensional space, the halo is white; if they are too close in the projection, the halo is dark. White was chosen for points that should be nearer because it stands out more, and while using the prototype ourselves, we often ended up looking for 'similar' samples.

Size and brightness were chosen over colour or shape, as using colour would have clashed with the colouring of the dots, and different shapes were not as glanceable as changes in brightness.

4.5.2 Distance correction

After examining the approximation errors, the viewer might decide that the errors of a certain point warrant more attention. They can then visualize them by selecting to view the high-dimensional distances between the selected point and all others. This removes all projection errors when it comes to distance, but for the selected sample only. The new, corrected positions for each point are calculated by taking the vector between the selected and the other point and multiplying it by the distance error ratio between the two. The angle between them is kept as is. As a result, the other point moves directly towards or away from the specified point, correcting the distance.

[J. Stahnke et al., 2015]
Showing Projection Errors

White: higher levels of similarity
Gray: lower levels of similarity

[J. Stahnke et al., 2015]
User Study & Results

• Types of Questions:
  - How would you try to characterize the type X?
  - In what way are X and Y different in their properties?
  - Are the projections of X and Y correct or do they deviate? How do you interpret this?
  - Can you discover which parts of the cluster combinations are A, B, and C?

• Discussion:
  - Learnability: need more effective mechanisms for grasping the concepts behind dimensionality reduction
  - Manipulation: What happens with results?
  - Large data: What about text corpora?

[J. Stahnke et al., 2015]
Focus+Context

- Show everything at once but compress regions that are not the current focus
  - User shouldn't lose sight of the overall picture
  - May involve some aggregation in non-focused regions
  - "Nonliteral navigation" like semantic zooming
- Elision
- Superimposition: more directly tied than with layers
- Distortion
Focus+Content Overview

- **Embed**
  - Elide Data
  - Superimpose Layer
  - Distort Geometry

- **Reduce**
  - Filter
  - Aggregate
  - Embed

[Munzner (ill. Maguire), 2014]
Elision

- There are a number of examples of elision including in text, DOI Trees, …
- Includes both filtering and aggregation but goal is to give overall view of the data
- In visualization, usually correlated with focus regions
Degree of Interest Function

- DOI = I(x) - D(x,y)
  - I: interest function
  - D: distance (semantic or spatial)
  - x: location of item
  - y: current focus point (could be more than one)

- Interactive: y changes
Elision: DOI Trees

- Example: 600,000 node tree
  - Multiple foci (from search results or via user selection)
  - Distance computed topologically (levels, not geometric)

[Heer and Card, 2004]
Superimposition

• Different from layers because this is restricted to a particular region
  - For Focus+Context, superimposition is not global
  - More like overloading

• Lens may occlude the layer below
Superimposition with Interactive Lenses

(a) Alteration

(b) Suppression

[ChronoLenses and Sampling Lens in Tominski et al., 2014]
Superimposition with Interactive

(c) Enrichment

[Extended Lens in Tominski et al., 2014]
It can be difficult to observe micro and macro features simultaneously with complex graphs. If you zoom in for detail, the graph is too big to view in its entirety. If you zoom out to see the overall structure, small details are lost.

Focus + context techniques allow interactive exploration of an area.

Mouseover to distort the nodes.

-D. Koop, DSC 530, Spring 2018
Distortion Choices

• How many focus regions?
  - One
  - Multiple

• Shape of the focus?
  - Radial
  - Rectangular
  - Other

• Extent of the focus
  - Constrained similar to magic lenses
  - Entire view changes

• Type of interaction:
  - Geometric, moveable lenses, rubber sheet
Overplotting
Cartesian Distortion
Stretch and Squish Navigation

Figure 3. LiveRAC shows a full day of system management time-series data using a reorderable matrix of area-aware charts. Over 4000 devices are shown in rows, with 11 columns representing groups of monitored parameters. (a): The user has sorted by the maximum value in the CPU column. The first several dozen rows have been stretched to show sparklines for the devices, with the top 13 enlarged enough to display text labels. The time period of business hours has been selected, showing the increase in the In pkts parameter for many devices. (b): The top three rows have been further enlarged to show fully detailed charts in the CPU column and partially detailed ones in Swap and two other columns. The time marker (vertical black line on each chart) indicates the start of anomalous activity in several of spire’s parameters. Below the labeled rows, we see many blocks at the lowest semantic zoom level, and further below we see a compressed region of highly saturated blocks that aggregate information from many charts.

Principle: multiple views are most effective when coordinated through explicit linking. The principle of linked views [15] is that explicit coordination between views enhances their value. In LiveRAC, as the user moves the cursor within a chart, the same point in time is marked in all charts with a vertical line. Similarly, selecting a time segment in one chart shows a mark in all of them. This technique allows direct comparison between parameter values at the same time on different charts. In addition, people can easily correlate times between large charts with detailed axis labels, and smaller, more concise charts.

Assertion: showing several levels of detail simultaneously provides useful high-information density in context. Several technique choices are based on this assertion. First, LiveRAC uses stretch and squish navigation, where expanding one or many regions compresses the rest of the view [11, 17]. The accompanying video shows the look and feel of this navigation technique. The stretching and squishing operates on rectangular regions, so expanding a single chart also magnifies the entire row for the device it represents, and the entire column for the parameters that it shows. The edges of the display are fixed so that all cells remain within the visible area, as opposed to conventional zooming where some regions are pushed off-screen. There are rapid navigation shortcuts to zoom a single cell, a column, an aggregated group of devices, the results of a search, or to zoom out to an overview. Users can also directly drag grid lines or resize freely drawn on-screen rectangles. Navigation shortcuts can also be created for any arbitrary grouping, whose cells do not need to be contiguous. This interaction mechanism affords multiple focus regions, supporting multiple levels of detail.

Second, charts in LiveRAC dynamically adapt to show visual representations adapted in each cell to the available screen space. This technique, called semantic zooming [13], allows a hierarchy of representations for a group of device-parameter time-series. In Figure 3, the largest charts have multiple overlaid curves and detailed axis and legend labels. Smaller charts show fewer curves and less labeling, and at smaller sizes only one curve is shown as a sparkline [24]. On each curve, the maximum value over the displayed time period is indicated with a red dot, the minimum with a blue dot, and the current value with a green one. All representation levels color code the background rectangle according to dynamically changeable thresholds of the minimum, maximum, or average values of the parameters within the current time window. The smallest view is a simple block, where this color coding is the only information shown.

Third, aggregation techniques achieve visual scalability by ensuring dense regions show meaningful visual representations. Given our target scale of dozens of parameters and thousands of devices, the size of the matrix could easily surpass 100,000 cells. Stretch and squish navigation allows users to quickly create a mosaic with cells of many different sizes.

[McLachlan et al., 2008]
The concerns we discuss in this paper emerged in the design and evaluation of fisheye interfaces that aim to support programming [21, 23]. With the specific goal of helping programmers navigate and understand source code, we have integrated a fisheye view with the Java editor in Eclipse, an open source development platform.

Basically, the fisheye view works by assigning a degree of interest (DOI) to each program line based on its a priori importance and its relation to the user’s current focus in the file. Then, lines with a DOI below a certain threshold are diminished or hidden, resulting in a view that contains both details and context.

Below, we discuss the fisheye interface design used in an initial controlled experiment [21], and the design used in a later field study [23], arguing for the changes made to the initial design.

Fig. 2. The fisheye interface initially studied [21] contains an overview of the entire document shown to the right of the detail view of source code. The detail view is divided into a focus area and a context area (with pale yellow background color) that uses a fixed amount of space above and below the focus area. In the context area, program lines that are less relevant given the focus point are diminished or hidden.[Jakobsen and Hornbaek, 2011]
Distortion vs. Hide

Fig. 3. The fisheye interface evolved for use in a field study [23]. Less interesting lines are hidden in the context area by using a magnification factor of 0. However, all lines with a degree of interest above a given threshold are included in the context area. In the example shown here, the bottom context area contains more lines than can be shown simultaneously. The context can be scrolled to view lines that are not initially shown. The motivation for this change is that all the lines may be important to the user. This design thus aims to guarantee users that the context area contains all the lines they expect to find (e.g., all occurrence so far a variable has selected).

3.3 Findings from User Studies

Overall, the results from our studies attest to the usefulness of fisheye interfaces to programmers. Participants in a controlled experiment preferred the fisheye interface to a linear source code interface [21]. Participants in a field study adopted and used the fisheye interface regularly and across different activities in their own work for several weeks [23]. The fisheye interface does not seem useful in all tasks and activities, however. Participants in the experiment completed tasks significantly faster using the fisheye interface, a difference of 10% in average completion time, but differences were only found for some task types. Although the results indicate usability issues, they also suggest that some tasks were less well supported by the fisheye interface. In addition, data from the field study showed periods where programmers did not use the fisheye interface, and debugging and writing new code were mentioned as activities for which the fisheye interface was not useful.

[Jakobsen and Hornbaek, 2011]
Research Questions

- Is a priori importance useful (and for what)?
- What does the user focus on?
  - predictability of view changes when focus changes
  - how direct user control is
  - task & context
- What interesting information should be displayed
  - degree of interest function may produce varied result sizes
- Do fisheye views integrate or disintegrate?
  - interference with other interactions; allow on-demand use?
- Are fisheye views suitable for large displays?

[Jakobsen and Hornbaek, 2011]
Distortion Concerns

• Distance and length judgments are **harder**
  - Example: Mac OS X Dock with Magnification
  - Spatial position of items changes as the focus changes

• Node-link diagrams not an issue… why?

• Users have to be made aware of distortion
  - Back to scatterplot with distortion example
  - Lenses or shading give clues to users

• **Object constancy**: understanding when two views/frames show the same object
  - What happens under distortion?
  - 3D Perspective is distortion… but we are well-trained for that

• Think about **what** is being shown (filtering) and method (fisheye)
H3 Layout

Large Graph Exploration with H3Viewer and Site Manager

(Demo)

[T. Munzner, 1998]
H3 Layout

Large Graph Exploration with H3Viewer and Site Manager

(Demo)
Focus+Context in Network Exploration

(a) Moderately large graph drawn with straight line edges. The graph nodes correspond to the USA major cities; edges show migration flows. The graph contains 1715 nodes and 9778 edges. Nodes are laid out according to geographical positions of cities, producing a drawing with poor readability, where edges mix in a totally unordered way and where some nodes are close to unnoticeable.

(b) The same graph as in Fig. 1(a) now drawn using edge bundling with edges rendered as B´ezier curves.

Figure 1: Illustration of edge bundling.

(a) The fish-eye distorts a small region of the graph for local inspection.
(b) The magnifying lens shows a zoom on a local region.

Figure 2: Fisheye and magnifying lens.

Magnifying Lens and Fish-eye – The magnifying lens \[3\] and geometrical fish-eye \[7\] were also added to the system as basic interactors. They allow to get local details on an area of the graph without having to zoom in (see Fig. 2(a) and Fig. 2(b)). These techniques allow to get an estimate on the degree of node or number of edges that have been bundled together, and an idea on the spatial organization of neighborhoods.

Neighborhood highlighting – After edges have been bundled, the graph gains in overall readability at the loss of more local information. For instance, connections between any two particular nodes cannot be easily recovered and isolated out of a bundle. When designing the system and deciding on the interactions to implement and combine, we focused on the recovery of these local information. By hovering the mouse over any node in the graph drawing, the user can highlight its neighborhood. This is accomplished by showing a translucent circle over the immediate area where a node sits while clearly displaying the neighborhood of the node (top of Fig. 3(a)). The circle fades off nodes not belonging to the selected neighborhood, temporarily providing a clear view of it. The size of the translucent circle is fitted as to enclose all immediate neighbors of the node in the graph. Using the mouse wheel, the user can select neighbors sitting at a bounded distance from the node. The size of the translucent circle adjusts accordingly (bottom of Fig. 3(b)).

Bring & Go – Now, neighborhoods in the graph don’t always sit close. As a consequence, the translucent circle highlighting neighbors of a node can potentially be quite large. That is, the distance between nodes in the graph does not always match their Euclidean distance in the drawing – [Lambert et al., 2010].
Focus+Context in Network Exploration

(a) Neighborhood highlighting – selecting a node brings up its neighbors, fading away all other graph elements.
(b) Using the mouse wheel, the neighborhood is extended to nodes sitting further away.

Figure 3: Illustration of the Neighborhood highlighting in interaction

This indeed is the challenge posed to all layout algorithms. The Bring & Go technique introduced by Tominski et al. [18] solves this paradox. The Bring operation pulls neighbors of a node to near proximity, temporarily resolving a situation where the layout algorithm had failed. Fig. 4(a) and Fig. 4(b) illustrates this situation – the passage from step 1 to step 2 being smoothly animated. Once the neighbors have been repositioned close to the node, the Go operation lets the user decide of a new direction to move to by selecting a neighbor. After clicking a neighbor node, the visualization is panned until re-centered around the target neighbor. The transition is performed by smoothly animating the pan (see Fig. 3). A recent user-study of this interaction technique has been made by Moscovich et al. [15]. When bringing neighbors close to the selected node, the edges abandon their curve shapes and are morphed to straight lines. This is done by modifying the control points coordinates of each curve so that they are all aligned.

Our system thus comprises a comprehensive palette of interactions focusing on adjacency or accessibility tasks (we borrow this terminology from Lee et al.'s [14] task taxonomy, itself referring to the work of Amar et al. [1]). That is, tasks such as exploring neighbor nodes, or counting them, finding how many nodes can be accessed from any given one, etc., can be easily done through direct manipulation of the graph using zoom, pan, neighborhood highlight or Bring & Go, for instance. All these interactions techniques have been implemented as interactor plugins for the Tulip graph visualization software [2] and are available through its plugin server.

4 Maintaining fluid interaction

The challenge we were faced with is that curves generation have a relatively high computational cost when it comes to interacting with bundles. Indeed, although the curves can be drawn in reasonable time for static drawings using standard rendering techniques, the problem becomes tedious when one wants to interact on bundles using any of the techniques described in the previous section. The curves' shapes must be continually transformed as the user moves the mouse and pilots interaction (geometrical fisheye or Bring & Go for instance).

Moreover, we did not want fluidity to impact on the quality of the curves and impose an upper bound on the number of control points used to compute the edge routes. Instead, we aimed at producing a system capable of dealing with an arbitrary number of control points. As a consequence, the computation of the points interpolating the curve itself puts a real burden on the system and calls for an extremely efficient approach. The solution we designed avoids performing computations on the CPU as far as possible, relying on the GPU for almost all curve related computations. The only computations that are potentially performed on the CPU are the original graph layout and the bundling part.

4.1 Introduction to spline rendering

Now, there are two major issues when rendering a parametric spline. Control points define the curve analytically described as a polynomial (see Eq. (1 for Bézier curves). Second, once the polynomial has been determined, it must be evaluated as many times as required in order to interpolate the curve itself. As a consequence, when interacting with the graph asking for local deformation of edges, bringing neighbors closer or following an edge, the curves must be re-computed on the fly.

A classical approach when rendering a curve is to compute the interpolation points on the CPU, then call appropriate graphics primitives and let the GPU render the curve. [Lambert et al., 2010]
Focus+Context in Network Exploration

(a) Bring (step 1) – Selecting a node fades out all graph elements but the node neighborhood.
(b) Bring (step 2) – Neighbor nodes are pulled close to the selected node.
(c) Go – After selecting a neighbor (the green node in Fig. 4(b)), a short animation brings the focus towards a new neighborhood.

Figure 4: Illustration of the Bring & Go interaction.

4.2 GPU-intensive spline rendering

Our solution delegates the computation of curve points to the GPU which is perfectly well designed to perform vectorial computation and floating points operations. By using the OpenGL Graphics API, we can encapsulate those tasks in a shader program. This type of program, written in a C-like language called GLSL (OpenGL Shading Language), allows to modify the default behavior of some processing units in the rendering pipeline – the vertex processing unit can be customized this way. The purpose of vertex processing stage is to transform each vertex’s 3D position in virtual space to the 2D coordinates at which it appears on the screen. By designing a vertex shader we can manipulate properties such as node position or color, with all computations executed on the GPU. Shaders offer tangible benefits since they are well suited for parallel processing as most modern GPUs have multiple shader pipelines.

The vertex shader we designed is activated each time we render a curve on screen. Before sending vertex coordinates to the GPU, the curve’s control points are transferred to the shader and stored in an array. The maximum size of that array is hardware dependent and determined at runtime. On recent GPU, more than one thousand control points can be used. Shaders are able to process a large number of control points, allowing to draw complex curves efficiently. However, most of the OpenGL implementations have restrained the maximum authorized number of control points to eight. So to draw a Bézier curve or a cubic B-spline with more than eight control points using evaluators, it has to be done piecewise by subdividing the curve to render into curves with fewer control points. Consequently, the performance to draw high order curves with this technique decreases as the number of control points grows. So even if evaluators work well to render curves with a small number of control points, they are not suitable to resolve our issue of drawing curves with several dozens of control points efficiently.