Data Visualization (DSC 530/CIS 602-01)

Focus+Context & Sets

Dr. David Koop
Multiple Views

- **Juxtapose and Coordinate Multiple Side-by-Side Views**
  - **Share Encoding: Same/Different**
    - *Linked Highlighting*
  - **Share Data: All/Subset/None**
  - **Share Navigation**

[Munzner (ill. Maguire), 2014]
## Multiple Views

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Data</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All</td>
<td>Subset</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Same</td>
<td>Redundant</td>
<td>Overview/Detail</td>
<td>Small Multiples</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Different</td>
<td>Multiform</td>
<td>Multiform, Overview/Detail</td>
<td>No Linkage</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Munzner (ill. Maguire), 2014]
Multiple Views

Partition into Side-by-Side Views

Superimpose Layers

[Munzner (ill. Maguire), 2014]
Multiform

[Improvise, Weaver, 2004]
Small Multiples

• Same encoding, but different data in each view (e.g. SPLOM)

[http://bl.ocks.org/mbostock/4063663]
Overview-Detail View

[Wikipedia]
Figure 2: Grouped Bar Chart

D. Koop, DSC 530, Spring 2016

[M. Bostock, http://bl.ocks.org/mbostock/3887051]
Example: Small Multiples Bar Chart

Group 1

Q108

Q208

Q308

Q408

Q109

Q209

Q309

Q409

Group 2

Group 3

Group 4

Example: Small Multiples Bar Chart

[M. Bostock, http://bl.ocks.org/mbostock/4679202]
Example: Trellis Matrix Alignment

In Figure 2 there are 6 panels, 1 column, 6 rows, and 1 page. Later, we will show a Trellis display with more than one page. We refer to the rectangular array as the trellis because it is reminiscent of a garden trelliswork.

Each panel of a trellis display shows a subset of the values of panel variables; these values are formed by conditioning on the values of conditioning variables. In Figure 1 the panel variables are variety and yield, and the conditioning variables are site and year.

![Trellis Matrix Alignment Diagram](image_url)

[Becker et al., 1996]
Example: HiVE System

[Slingsby et al., 2009]
Example: Superimposed Line Charts

[M. Bostock, http://bl.ocks.org/mbostock/3884955]
Overview

Reducing Items and Attributes

- **Filter**
  - Items
  - Attributes

- **Aggregate**
  - Items
  - Attributes

Reduce

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

[Munzner (ill. Maguire), 2014]
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.

By JEREMY WHITE

Source: New York City Department of Health and Mental Hygiene

© 2013 The New York Times Company

Gracie's Cafe
Grade pending
Violation points 27
Click for details
Attribute Filtering on Star Plots

(a)                 (b)

(c)                 (d)

[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

[Munzner (ill. Maguire), 2014]
Boxplots

- Show **distribution**
- Single value (e.g. mean, max, min, quartiles) doesn't convey everything
- Created by John Tukey who grew up in New Bedford!
- Show **spread** and **skew** of data
- Best for **unimodal** data
- Variations like vase plot for multimodal data
- Aggregation here involves many different marks
Spatial Aggregation

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

in cartography, changing the boundaries of the regions used to analyze data can yield dramatically different results.
Spatial Aggregation

Spatial aggregation is a technique in cartography where the boundaries of regions used to analyze data are modified. Changing the boundaries of the regions can yield dramatically different results.

[Penn State, GEOG 486]
Modifiable Areal Unit Problem

- How you draw boundaries impacts the type of aggregation you get
- Similar to bins in histograms
- Gerrymandering

[Wonkblog, Washington Post, Adapted from S. Nass]
Exam 2

- [http://www.cis.umassd.edu/~dkoop/dsc530-2016sp/exam2.html](http://www.cis.umassd.edu/~dkoop/dsc530-2016sp/exam2.html)
- Format: Multiple Choice & Short Answer
- Coding Questions:
  - Understand HTML, CSS, SVG, and JavaScript
  - Any code writing will be in pseudocode
- Research papers on sets
- Example Questions:
  - Identify a proper colormap for an attribute or set of attributes
  - What is a requirement when doing map visualization to generate a two-dimensional figure?
  - Identify the strategy used to address complexity in the given visualization (e.g. juxtaposition, partition, superimposition).
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)

Gene 1
Gene 2
Gene 3

original data space

Gene 2
Gene 1

Component space

PC 1
PC 2

Gene 3

PC 2
PC 1

PCA

[M. Scholz, CC-BY-SA 2.0]
Non-linear Dimensionality Reduction

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

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

original data space \( \mathcal{X} \)

component space \( \mathcal{Z} \)

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

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

Task 1

In HD data ➔ Out 2D data

What?

In High-dimensional data ➔ Out 2D data

Why?

Produce ➔ Derive

Task 2

In 2D data ➔ Out Scatterplot Clusters & points

What?

In 2D data

Why?

Discover ➔ Explore ➔ Identify

How?

Encode ➔ Navigate ➔ Select

Task 3

In Scatterplot Clusters & points ➔ Out Labels for clusters

What?

In Scatterplot Clusters & points

Why?

Produce ➔ Annotate

[Muñzner (ill. Maguire), 2014]
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

- 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, DOITrees, …

• Includes both filtering and aggregation but goal is to give overall view of the data

• In visualization, usually correlated with focus regions
Elision: DOI Trees

[Heer and Card, 2004]
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]
Distortion

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.

[M. Bostock, http://bost.ocks.org/mike/fisheye/]
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

[M. Bostock, http://bost.ocks.org/mike/fisheye/]
Cartesian Distortion

[M. Bostock, http://bost.ocks.org/mike/fisheye/]
Cartesian Distortion

[M. Bostock, http://bost.ocks.org/mike/fisheye/]
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 squish-ing 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]
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)
Focus+Context in Graph 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 Bézier 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 of the degree of nodes 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 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, neighborhood in the graph do not 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 –

[3] Lambert et al., 2010
Focus + Context in Graph 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 interaction 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.
Focus+Context in Graph 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.

on the screen. For instance, a Bézier curve corresponds to a polynomial whose degree is the number of control points determining it (other families of polynomials can also be used, such as Hermite’s polynomials). Let \((P_0, \ldots, P_n)\) be control points. The polynomial defined from these control points is:

\[
Q_n(t) = \sum_{i=0}^{n} B_{i,n}(t) P_i,
\]

where the sum is performed component wise and

\[
B_{i,n}(t) = \binom{n}{i} (1-t)^{n-i} t^i, \quad 0 \leq t \leq 1
\]

are Bernstein polynomials and \(\binom{n}{i} = \frac{n!}{i! (n-i)!}\) denotes the usual binomial coefficient.

In order to be able to easily interact with the edge bundled graphs, even for basic interactions like panning and zooming, we have to optimize the curves rendering by reducing the computational load on the CPU as much as possible. One solution could be to pre-compute all curve points and store them in memory; this obviously is not efficient in terms of memory usage, considering that we want to draw a large amount of fine-grained rendered curves. For example, drawing \(10^5\) curves (edges) with 100 points per curves – one point being stored as 3 floats (4 bytes each), the total amount of memory use would be \(\sim 10^8\) bytes (more than 110 Mbytes).

Another solution will be to use the built-in components of high level graphics API for rendering curves. For instance, in OpenGL, that task can be achieved by using a standard feature called evaluators. Evaluators can be used to construct curves and surfaces based on the Bernstein basis polynomials. This includes Bézier curves and patches, and B-splines. An evaluator is set up from an array of control points and allows to compute curve points on the GPU by sending the parameter \(t\) to the rendering pipeline. 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.

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 stored in one array.
## Dataset Types

<table>
<thead>
<tr>
<th>Tables</th>
<th>Networks &amp; Trees</th>
<th>Fields</th>
<th>Geometry</th>
<th>Clusters, Sets, Lists</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>Items (nodes)</td>
<td>Grids</td>
<td>Items</td>
<td>Items</td>
</tr>
<tr>
<td>Attributes</td>
<td>Links</td>
<td>Positions</td>
<td>Positions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Attributes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- How do we visualize sets?
- What about considering text documents as datasets?
Sets

- A set is an unordered group of items (S = {"apple", "pear", "orange"})
- What questions can we ask about sets?
  - Containment: Is some item x in S?
  - Intersection: what items are in both S and T?
  - Union: what items are in either S or T?
  - Difference: what items are in S but not T?
  - …
Venn Diagram

[http://askville.amazon.com/idea-Venn-diagram/AnswerViewer.do?requestId=8420613]
Venn Diagram?

[http://viz.wtf/post/95470812927/this-deserves-an-all-time-high-score]
Scalability

- How to show the intersection of four sets? 8?
- Euler Diagrams: only show intersections/containments that exist
- Still run into scalability issues
MANUSCRIPT RECEIVED 31 MARCH 2010; ACCEPTED 1 AUGUST 2010; POSTED ONLINE 24 OCTOBER 2010; MAILED ON 16 OCTOBER 2010.

1

WE IDENTIFY TWO MAIN CHALLENGES WHEN DRAWING EULER DIAGRAMS:

- Sets are not required to overlap if their corresponding intersection is empty.
- The relaxation of Venn diagrams in which the shapes corresponding to sets intersect in complex ways, this type of representation becomes a challenging problem in information visualization.
- The common visual representation of sets are Venn and Euler style diagrams.
- Complexity of set regions plays a key role in perception [23] and in our ability.
- Intersect in complex ways, this type of representation becomes a challenging problem in information visualization.
- The topic of this paper is visual representations of data elements such as communities and study their relationships. There is a wide range of analysis scenarios. For example, when analyzing documents, linguists often group words into semantic categories and topics. Similarly, when analyzing social networks, sociologists group people into communities and study their relationships. There is a wide range of analysis scenarios. For example, when analyzing documents, linguists often group words into semantic categories and topics. Similarly, when analyzing social networks, sociologists group people into communities and study their relationships.

Abstract

In many common data analysis scenarios the data elements are logically grouped into sets. Venn and Euler style diagrams are a common visual representation of such set membership where the data elements are represented by labels or glyphs and sets are represented as non-convex regions as well as placing labelled elements inside these regions automatically. They demonstrate how their techniques to compute sets (or clusters) based on similarity data [22].

Index Terms

- Information Visualization
- Euler diagrams
- Set Visualization
- Graph Visualization

Introduction

The common visual representation of sets are Venn and Euler style diagrams. Venn diagrams represent all sets and their possible intersections with overlapping elliptical shapes. Euler diagrams are a relaxation of Venn diagrams in which the shapes corresponding to sets are required to be continuous and convex. Several approaches exist to draw such set regions using more complex shapes. However, the resulting diagrams can be difficult to interpret. In this paper we present two novel approaches for simplifying a complex diagram.

Compact Rectangular Euler Diagrams

Fig. 1. Compact Rectangular Euler Diagram(left) and Euler Diagram with Duplications(right)

- Compact Euler Diagrams
- Untangling Euler Diagrams
- Tim Dwyer is with Microsoft Corp., E-mail: timdwyer@microsoft.com.
Compact Euler Diagrams

[Image of a compact Euler diagram with names of individuals, such as Madison, Elizabeth, Jayden, Christopher, Daniel, and others.]
Bubble Sets & Related Techniques

- Given spatial layout is determined by other attributes, want to show set containment without modifying spatial layout
- Idea of "spatial rights"
- How?
Bubble Sets & Related Techniques

- Given spatial layout is determined by other attributes, want to show set containment without modifying spatial layout
- Idea of "spatial rights"
- How?
- Containment marks
Fig. 7: Grouping research articles on a timeline

Fig. 8: Items can be expanded to reveal a larger image or the article's abstract. The boundary moves to accommodate the larger item and other items move along the y-axis to remain visible and selectable.

Sets of hotels, subway entrances, and medical clinics may help them find a hotel that is central to several medical clinics and near a subway entrance.

4.4 Sets over Scatterplots

Scatterplots have clearly defined spatiality due to the numerical positioning of items. We add Bubble Sets to a reimplementation of the well-known GapMinder Trendalyzer. This scatterplot shows fertility rate against life expectancy and is animated over time. Data points represent countries, sized by population. Color and set membership is defined by the continent. The grouping of the Sub-Saharan Africa countries highlighted in Figure 6 reveals that while most of the countries in this set had high fertility rates and low life expectancies, there are two outliers: Mauritius and Reunion, which are islands in the Indian Ocean. As the data set includes data for many years, and since Bubbles Sets are calculated at interactive rates, the temporal changes can be convincingly shown through animation.

DISCUSSION AND FUTURE WORK

We have presented Bubble Sets, a method for automatically drawing set membership groups over existing visualizations with different degrees of requirements for primary spatial rights. In contrast to other overlaid containment set visualizations, Bubble Sets maximize set membership inclusion and minimize inclusion of non-set members. In fact, Bubble Sets can guarantee that all set members will be within one container, as opposed to the more common multiple disjoint containers. While Bubble Sets cannot guarantee non-set member exclusion, the routing algorithm minimizes these occurrences.

Within our isocontour approach, we have implemented several heuristics to reduce surface calculation and rendering time, such as grouping pixels for potential calculations and restricting the regions in which items influence the potential field. The current implementation works without noticeable lag; items can be dragged and the surface follows for our examples order of 655 nodes, 65–85 sets. For example, it takes on average 65 ms to calculate the virtual edge set, fill the energy field, find the contour, and render the Sub-Saharan Africa set in a window size 6–85 pixels. That set has 0 items and the entire scatter plot has 6–: points. The majority of this time is spent creating the virtual edge set. An incremental approach, using A*-search as in [89], may provide improvements in speed and stability.

As the number of items, the screen resolution, or the number of sets increases, so will the rendering time. Additional techniques, such as grouping close items into larger pseudo-nodes, and caching the energy field values between frames may increase the capacity of the system.

ACKNOWLEDGMENTS

Thanks to the following organizations for supporting this research: iCORE, CFI, NSERC, and SMART Technologies.
We performed a controlled experiment with 13 participants, comparing KelpFusion to Bubble Sets and LineSets. Users preferred KelpFusion's response times, accuracy, and ability to capture the intended region. KelpFusion's visualizations adapt to point sets of varying density, providing a good sense of grouping and aesthetically pleasing results for users. 

Figure 1. Visualizations using the various methods discussed in this paper. (a) Image generated using the implementation generously provided by the authors of Bubble Sets [7]. (b) Image courtesy of Kasper Dinkla. (c) Image generated using the LineSets implementation. (d-f) Images generated by our KelpFusion implementation. 

[Meulemans et al., 2013]

ELATED WORK

Venn or Euler diagrams are popular ways to visually represent set intersections. In these diagrams, closed curves correspond to sets and overlaps between the curves indicate intersections. Several papers have explored the problem of automatically drawing Euler diagrams to convey abstract set relationships. For example, Simonetto and Auber [14] described a method for generating Euler diagrams. Other approaches investigated the set topology. 

KelpFusion: Bubble Sets, LineSets, and Kelp Diagrams, in comparison to a shortest-path graph, enable interactive manipulation of the proximity graph and its boundary. Points are spatially close if they are likely inside the intended region [2]. Shortest-path graphs constructed the boundary of a region based on points that are close. 

We showed that KelpFusion outperformed Bubble Sets in accuracy and completion time. User preferences and comments also indicated that KelpFusion was on par with LineSets in terms of accuracy but yielded faster completion time. We found that KelpFusion was on par with Bubble Sets in accuracy but yielded faster completion time. We also found that KelpFusion was on par with Bubble Sets in accuracy but yielded faster completion time.
Parallel Sets

[Kosara et al., 2006, Example: J. Davies]