Visualizing Multidimensional Query Results Using Animation

Amit P. Sawant[a] and Christopher G. Healey[b]

[a, b]North Carolina State University, Department of Computer Science, Raleigh, NC, USA

ABSTRACT

Effective representation of large, complex collections of information (datasets) presents a difficult challenge. Visualization is a solution that uses a visual interface to support efficient analysis and discovery within the data. Our primary goal in this paper is a technique that allows viewers to compare multiple *query results* representing user-selected subsets of a multidimensional dataset. We present an algorithm that visualizes multidimensional information along a space-filling spiral. Graphical glyphs that vary their position, color, and texture appearance are used to represent attribute values for the data elements in each query result. Guidelines from human perception allow us to construct glyphs that are specifically designed to support exploration, facilitate the discovery of trends and relationships both within and between data elements, and highlight exceptions. A clustering algorithm applied to a user-chosen ranking attribute bundles together similar data elements. This encapsulation is used to show relationships across different queries via animations that morph between query results. We apply our techniques to the MovieLens recommender system, to demonstrate their applicability in a real-world environment, and then conclude with a simple validation experiment to identify the strengths and limitations of our design, compared to a traditional side-by-side visualization.

Keywords: animation, color, information visualization, motion, perception, scientific visualization, texture

1. INTRODUCTION

Visualization is an area of computer graphics that presents information in a visual form to facilitate rapid, effective, and meaningful analysis and interpretation. Example application domains include scientific simulations, land and satellite weather information, geographic information systems, and molecular biology. Visualization is also used in more abstract settings, for example, software engineering, data mining, and network security. A key challenge is designing visualizations that are effective for the user's data and analysis tasks. Our approach constructs visual representations that harness the strengths of the low-level human visual system. These perceptual visualizations display the data in ways that allow items of interest to capture the user's focus of attention.

Due in large part to a rapid increase in the size of the average dataset, the time per element needed to analyze the data is often critical. The desire to extract knowledge efficiently motivates the need for an effective visualization system. A dataset's size is made up of three related properties: (1) the number of elements stored in the dataset; (2) the number of attributes represented within the dataset; and (3) the range of values possible for each attribute. The information content of a visualization is a combination of these same properties: the number of elements, the number of attribute values per element, and the range of different attribute values being visualized. The new technique described in this paper seeks to increase information content by focusing on the last two properties, dimensionality and range.

In our work, "query results" refers to collections of data elements that form subsets of the original dataset. How the results are generated is independent of our visualization algorithm. They can come from direct queries into a dataset (e.g., via SQL queries on a relational database), or from other methods of data filtering or data selection (e.g., via mathematical, spatial, or temporal filters applied to a scientific dataset). Based on a common need to visualize relationships both within and between queries, we identified three important goals for our research:

1. Design graphical glyphs that support flexibility in their placement, and in their ability to represent multidimensional data elements.
2. Build effective visualization techniques that uses the glyphs to represent results from queries on a multidimensional dataset.

Further author information: Amit P. Sawant, E-mail: amit.sawant@ncsu.edu, Christopher G. Healey, E-mail: healey@csc.ncsu.edu

3. Highlight similarities and differences between different queries using animation.

Our intent is to show not just the intersection of multiple query results, but also specific details about similarities and differences in the underlying structure of the results. This information can be critical to understanding how the original queries are themselves related. To our knowledge, the combination of multidimensional display techniques, perception, and animation for direct comparison of different perspectives into a dataset represents a useful and novel contribution to the field.

1.1. Design

Our visualization algorithm consists of two basic stages:

1. Select effective visualizations for individual query results.
2. Visualize correspondence between query results.

We display query results using a data-feature mapping that defines a visual feature to use to represent each data attribute. The mapping controls the appearance of a geometric glyph through the attribute values of the data element it visualizes. Our mappings are built using experimental results that describe how the human visual system perceives different color, texture, and motion properties. Glyphs are positioned along a spiral embedded in a plane based on a scalar ranking attribute. The value of the ranking attribute decreases as glyphs move away from the center of the spiral.

Correspondence between queries is represented with animations that morph between pairs of query results. A clustering algorithm identifies groups of spatially neighboring glyphs that are common across the queries. This technique allows a viewer to see both the similarities and the differences between different query results. Groups of glyphs are moved in sequence to highlight elements that maintain their relative ranking with one another. This is important, since ranking is assumed to be an attribute of significant interest to the viewer.

We used the MovieLens recommender system (`http://movielens.umn.edu`) as a practical testbed for our techniques.[1] Graduate students in our laboratory were asked to rate movies that they have seen. Based on this information, MovieLens matches their ratings with other users in the database. The resulting profile allows MovieLens to suggest movies that the students have not seen, but would probably like. We used these recommendations, together with a variety of additional information about each movie, as input to our visualization system. This allows students to see their recommendations, and to compare them with recommendations for their friends. We conducted a simple validation experiment within this real-world environment, to compare our animated design to a more traditional side-by-side visualization of different recommendations. Results showed that our design was as good or better than a side-by-side visualization for four of the five tasks we studied.

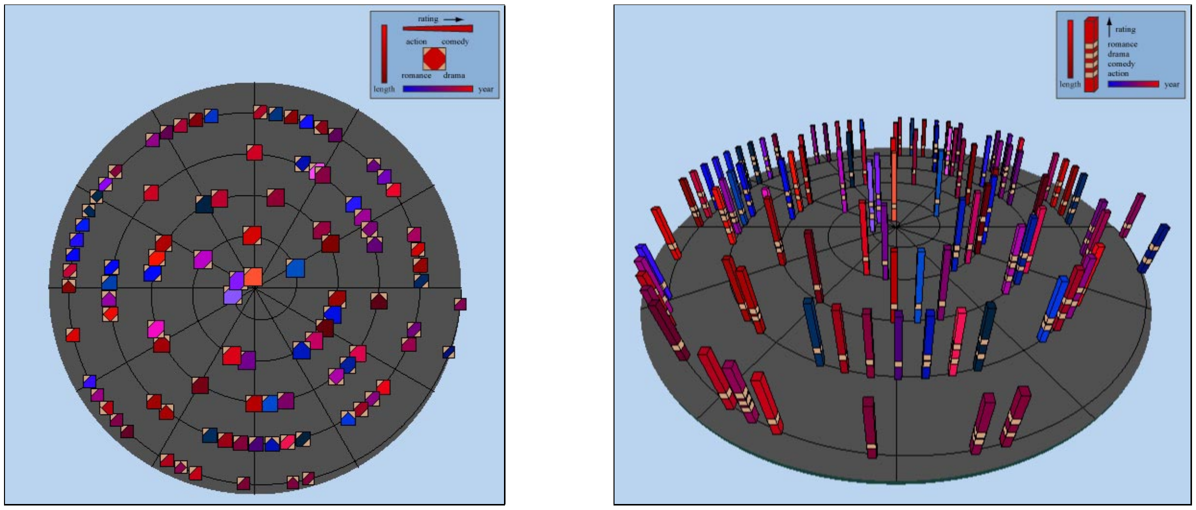
2. INFORMATION VISUALIZATION

The basic steps in visualizing information are: (1) convert the raw data to a data table; (2) map the data table to a visual structure; (3) apply view transformations to increase the amount of data that can be visualized; and (4) allow user interaction with the mapping and the visual structures to create an information workspace for visual sense making.[2] A number of well-known techniques exist for visualizing non-spatial datasets. These can be roughly classified as geometric projection, iconic display, hierarchical, graph-based, pixel-oriented and dynamic.[3,4] We decided a dynamic iconic display was most relevant to our goal of visualizing relationships between multidimensional query results. An iconic display is used to represent patterns within each query result. A dynamic animation is used to highlight similarities and differences between queries.

2.1. Visualizing Along a Spiral

A one-dimensional ordering is imposed on the data elements through a user-selected scalar attribute, or "ranking" attribute. We needed a way to map this ordering to a 2D spatial position for each element. We chose to use a 2D space-filling spiral to satisfy this requirement. Our algorithm is based on a technique introduced by Carlis and Konstan to display data along an Archimedean spiral,[5] represented in its polar form as:

$$\left.\begin{array}{rcl} r &=& \alpha\theta \\ x &=& r\cos\theta \\ y &=& r\sin\theta \end{array}\right\} \; 0 \le \theta \le \theta_{\max} \qquad (1)$$



(a) (b)

Figure 1. Visualizing a MovieLens query using data-feature mapping M_{ML}: (a) in 2D; (b) in 3D

where θ tracks position along the spiral, r is the distance from the center of the spiral for a given θ, and α is a constant used to control the spacing between neighboring arcs in the spiral (Fig. 1). Visualizations along a spiral have been used with both abstract and periodic data (e.g., SpiralGlyphics[6]) to provide properties like: (1) comparison of values along a small section of the spiral; (2) comparison of several consecutive cycles of the spiral; (3) identification of periodic patterns in the data; and (4) changes in data patterns over the length of the spiral.[7]

Other space-filling curves exist, for example, Hilbert, Peano, or Sierpinski fractal curves. These curves are hierarchical, and can more efficiently fill the 2D plane. Because our curve is used to represent order, however, we must ensure that viewers can rapidly convert position on the curve to distance along the curve. This is an intuitive feature of a spiral (i.e., farther from the center implies farther along the curve), but is more difficult to achieve with other space-filling curves.

Some examples of circular visualization are DiskTree that represents hierarchical structures in a circular two-dimensional form,[8] Sunburst[9] that displays a radial visualization of information hierarchies, and MetaCrystal[10] that compares the results from multiple search engines. Zschimmer[11] used hierarchical circular visualization to display intranet search results.

Other spatial layouts exist, for example, a 2D scatterplot whose x and y-axes represent the ranking attributes for two separate query results. Elements unique to a single query result are visualized on the appropriate axis, while elements common to both query results appear in the body of the scatterplot. Although this technique is efficient and compact, extending it to compare more than two query results is complicated, requiring either a higher-dimensional scatterplot (e.g., a 3D volume for three query results), or a projection into 2D or 3D space prior to visualization.

3. PERCEPTION

Our data-feature mappings are constructed from psychophysical studies of how the visual system “sees” fundamental visual properties in an image. The use of color, texture, and motion has a long history in the graphics, vision, and visualization literature (e.g., in Treinish’s meteorological visualizations[12]).

Examples of simple color scales include the rainbow spectrum, red-blue or red-green ramps, and the grey-red saturation scale. More sophisticated techniques divide color along dimensions like luminance, hue, and saturation to better control the difference viewers perceive between different colors. Perceptually balanced color models have been combined with non-linear mappings to emphasize changes across specific parts of an attribute’s domain. Automatic colormap selection algorithms based on an attribute’s spatial frequency, continuous or discrete nature, and the analysis tasks to be performed have been proposed. Experiments have shown that color distance, linear separation, and color category must all be controlled to select discrete collections of distinguishable colors.[13]

Texture is often viewed as a single visual feature. Like color, however, it can be decomposed into a collection of fundamental perceptual dimensions. One promising approach in visualization is the use of perceptual texture dimensions to represent multiple data attributes. Individual values of an attribute control its corresponding texture dimension. The result is a texture pattern that changes its visual appearance based on data in the underlying dataset. Examples of perceptual dimensions include properties like size, density, orientation, and regularity of placement.[13,14]

A third visual feature we use is motion. Basic motion patterns are processed very rapidly by the low-level visual system.[15] Perceptual dimensions of motion like flicker, direction, and velocity have been studied in the psychophysical literature, and are now being used for notification in real-time systems,[16] for cognitive grouping of elements,[17] and for visualizing multiple independent data attributes.

We are particularly interested in applying motion through animations that visualize correspondence between query results. Researchers have demonstrated how animations can improve a visualization. Yee et al. explored trees by placing a user-selected node at the center of the screen, with child and parent nodes arranged in concentric circles around it.[18] As viewers choose new nodes to study, differences are introduced as a smooth, animated transition of nodes and edges from their current to their new locations. Robertson et al. animated relationships between user-selected nodes embedded within data hierarchies.[19] User studies confirmed that viewers' performance and preferences favored animation for transitioning between different hierarchies. Krasser et al. combined parallel coordinate plots with the animation of scatterplots to examine 2D and 3D coordinated displays that provide insight into network activity.[20]

Our visualization designs choose visual features that are highly salient, both in isolation and in combination. We map the features to individual data attributes in ways that draw a viewer's focus of attention to important areas in a visualization. The ability to harness the low-level human visual system is attractive, since:

- high-level exploration and analysis tasks are rapid and accurate, usually requiring 200 milliseconds or less to complete,
- analysis is display size insensitive, so the time to perform a task is independent of the number of elements in the display, and
- different features can interact with one another to mask information; psychophysical experiments allow us to identify and avoid these visual interference patterns.

We have combined our experimental results into a visualization assistant called ViA.[21] ViA employs mixed initiative AI techniques that use information about a dataset and a list of perceptual guidelines to search intelligently the space of all possible visualizations for the specific visualizations that are best-suited to the user's data and analysis needs. ViA suggested the initial data-feature mappings we used. These mappings were then extended to support an animated method of comparing pairs of query results.

4. ANIMATION

Animation in computer graphics is a sequence of static images taken at small, discrete intervals of time and displayed at a speed fast enough to form the impression of smooth motion. A more theoretical description defines animation as a change function used to produce a successor image from a current image.[7] Traditional computer animation techniques are common in visualization, for example, camera animations to zoom, spin, pan and track along a path inside a dataset. This type of animation is defined with the start and end positions of the camera, the path through the dataset, the number of frames to be displayed, the duration of the animation, the intermediate states to be highlighted in each step, and so on.[22] This same technique (with possibly fewer view parameters changing) is used to visualize temporal properties of a dataset. Finally, individual data elements can be animated to visualize the values of the attributes they encode.[16,23]

The static visualizations we produce represent data elements as glyphs whose color, texture, and position vary based on an element's specific attribute values. Our visualization can be viewed as a graph, where glyphs represent nodes and subsets of the spiral between pairs of glyphs represent edges. This formalism is useful, because it allows us to apply graph animation techniques to highlight the similarities and differences between pairs of visualizations.

5. VISUALIZATION DESIGN

Our visualizations were designed by first constructing an object to represent a single data element. Next, the objects are positioned to produce a static visualization of a single query result. Finally, an animation step is used morph between pairs of query results to highlight their similarities and differences.

5.1. Visualizing Data Elements

We designed both a 2D and 3D glyph (a square and a tower, respectively) to represent a data element (Fig. 1). A data-feature mapping built from our perceptual guidelines defines which visual feature to use to display values for a particular attribute. Our glyphs can vary different color and texture properties, including hue, luminance, size (or height for the 3D glyph), spatial density, and regularity of placement. A glyph uses the attribute values of the data element it represents to select specific values of the visual features to display.

There are many different objects we could have selected to represent our data elements, ranging from pixels,[4] through basic 2D shapes,[13, 14] and on to more complex 3D objects. Although 2D glyphs are "simpler", a 3D glyph may be better for representing multiple attribute values in certain situations, since it offers more surface area on which to place information. Psychophysical experiments have shown that viewers can properly identify and compare color and texture properties of 3D objects, as long as they are perceived as being displayed in a 3D environment.[13, 24] Visualizing glyphs on an underlying plane is used to reinforce this perception. Overlap or occlusion is a problem for both types of glyphs, particularly when many data elements are being visualized. Our glyphs vary their positions slightly to try to minimize their overlap with one another. The heights of the 3D glyphs decrease as they move away from the center of the spiral, further reducing the possibility of outer glyphs occluding inner ones.

5.2. Visualizing Query Results

Once glyphs are built for each data element in a query result, they must be positioned along an underlying spiral. The value of a user-selected scalar attribute A_r defines a glyph's position. Given $A_{r,\min} \leq a_r \leq A_{r,\max}$ for all $a_r \in A_r$, an element e_i with a ranking attribute value $a_{r,i}$ will have its glyph located at an (x,y) position defined by Eq. 1, where:

$$\theta = \frac{A_{r,\max} - a_{r,i}}{A_{r,\max} - A_{r,\min}} \theta_{\max} \tag{2}$$

In other words, elements with high ranking values are positioned near the center of the spiral, while elements with low ranking values are positioned near the periphery.

5.3. Animating Pairs of Query Results

We use animation to morph between pairs of query results Q_1 and Q_2. Our animations are designed to visually identify neighbors that maintain their relative positions (i.e., their relative ranking with one another) along the spiral. The animation consists of three phases. We begin by displaying results from Q_1 using our visualization technique for individual query results. We then gradually remove glyphs that are not present in Q_2. Next, the remaining glyphs that are common to both queries move to their new positions as defined by Q_2. Clusters of glyphs that maintain their spatial neighborhood with one another move first, followed by individual glyphs that are not part of any shared cluster. Each move is animated over 900 milliseconds (i.e., approximately one second). Finally, glyphs that are present only in Q_2 gradually appear.

We explored different ways to build the paths the glyphs follow as they move from their starting position to their ending position. In particular, we investigated: (1) straight linear paths, (2) angular polar paths, (3) angular B-Spline paths, and (4) equal arc B-Spline paths. A straight linear path (i.e., a linear interpolation of Cartesian coordinates) often crossed the spiral numerous times along its length. This can produce collisions between moving and stationary glyphs. Collisions are visually disruptive, with viewers sometimes being confused about which glyph entered and which glyph left a collision location. An angular polar path (i.e., a linear interpolation of polar coordinates) produced fewer collisions, but did not generate smooth animations. Interpolating angular values can produce non-uniform distances between steps, which results in non-uniform velocities along the animation path. We observed similar problems with angular B-Spline paths. We therefore chose to use an equal arc B-Spline path (i.e., a spline path divided into steps of uniform arc length), since it returns a smooth animation and minimizes collisions between glyphs.

5.4. Clustering

We applied a simple segmentation algorithm to partition a query result into clusters based on the user-chosen ranking attribute. All the elements in a cluster have "similar" ranking values. Two elements are said to be similar if the difference between their ranking attribute values is less than or equal to a user-chosen threshold ε.

Given ranking attribute A_r, our clustering algorithm proceeds as follows:

1. Array the elements e_i along the spiral based on their ranking attribute values $a_{i,r}$.
2. From the remaining elements that are not part of any cluster, choose the element e_1 closest to the center of the spiral.
3. Set a median ranking value m for the new cluster to $a_{1,r}$.
4. Choose the next non-clustered element e_i that is closest on the spiral to e_1. If $|a_{i,r} - m| \leq \varepsilon$ for some user-chosen threshold ε, add e_i to the cluster and update the cluster's median.
5. Continue until no more elements can be added to the cluster.
6. Repeat until every element in the query result has been assigned to a cluster.

The algorithm grows each cluster around a starting element. A new element e_i is accepted if its ranking value $a_{i,r}$ is within a user-chosen threshold ε to the cluster's median value m, $|a_{i,r} - m| \leq \varepsilon$. ε is fixed whereas m is updated as new elements are added. The granularity of the clusters depend on ε, and on the method used to update the median value. We use the median update method:

$$m = \frac{1}{\sum_{x=0}^{k-1} w^x} [w^0 a_{1,r} + w^1 a_{2,r} + \ldots + w^{k-1} a_{k,r}] \tag{3}$$

w is used to control how new elements affect the median value. On the range $0 < w < 1$, the maximum possible contribution of each new element is monotonically decreasing. Elements that are added to the cluster later in the process have a smaller and smaller influence on the median value. This allows m to stabilize after the initial elements are added, and ensures that the cluster cannot grow in size without bound. We used $w = \frac{7}{8}$ to produce a good balance between each new element having too much or too little impact on the median.

6. VISUALIZING MOVIE DATA

In order to test our system on a real dataset, we turned to the application that originally motivated our investigation: visualizing relationships within and between queries to the MovieLens recommender system.

6.1. Data Collection

MovieLens is a web-based recommender system for movies, built on collaborative filtering algorithms, and run by the GroupLens Research Group in the Department of Computer Science and Engineering at the University of Minnesota. MovieLens uses an experimental data source containing thousands of movies as a framework for studying user interface issues related to recommender systems.

MovieLens works as follows: users tell MovieLens about movies they have seen by providing a rating of each movie on a scale from one to five. One means a user did not like the movie at all, and five means a user liked the movie a lot. Once a sufficient number of ratings are provided, MovieLens builds a profile of a user's movie preferences. This profile is entered into a database, and matched against existing profiles to identify other users who share similar interests in movies. Once matches are found, MovieLens can recommend movies that the other users have seen and liked, but that the current user has not seen. This results in suggestions that are tailored to the personal tastes of a user. MovieLens includes a predicted *user rating* for each movie it recommends. This is MovieLens's estimate of how the current user would rate the movie if he or she had already seen it.

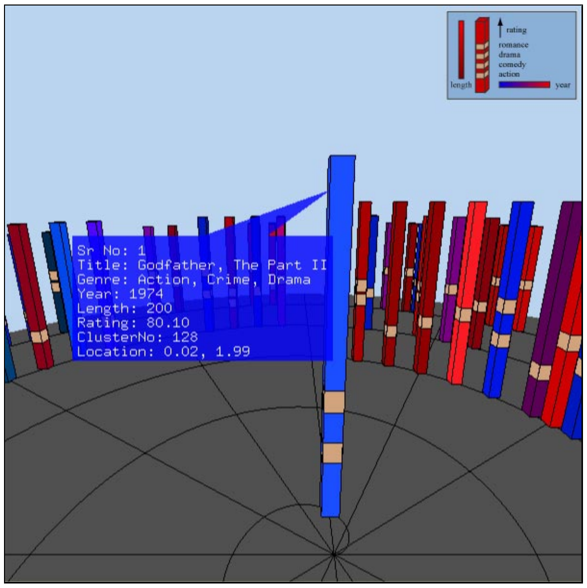
Graduate students from our laboratory volunteered to rate movies that they have seen. We used their profiles to produce multiple recommendations from MovieLens. MovieLens returns a movie's *title*, its *genre* and a predicted *user rating*. We augmented each recommendation with information taken from the Internet Movie Database[*] (IMDB) to include the *year* the movie was released, the *length* of the movie in minutes, and the *IMDB rating* of the movie.

MovieLens's predicted *user rating* falls along a nine-point scale ranging from 1 to 5 in $\frac{1}{2}$ point steps. We increased the granularity of this result by first extending MovieLens's predicted *user rating* to the range 1 to 10, then weighting it by the *IMDB rating*. The *IMDB rating* is the average of all IMDB users who have rated a particular movie. It ranges from 1 to 10 in $\frac{1}{10}$ point steps. The result is a predicted *user rating* over the range 1 to 100, with values rounded to produce a total of 1000 discrete steps.

6.2. Data-Feature Mappings

The query results we used as input for our visualization system have the following attributes: movie *title*, *genre* of the movie, *year* the movie was released, *length* in minutes, and predicted *user rating*. Each movie recommendation is displayed as a 2D or 3D glyph. Based on suggestions from ViA, our MovieLens data feature mapping M_{ML} defined:

- Predicted *user rating* mapped to size for 2D glyphs (small for low ratings to large for high ratings), or height for 3D glyphs (short for low ratings to tall for high ratings).
- *Year*, which ranges from 1921 to 2006, mapped to hue (blue for oldest to red for most recent).
- *Length* mapped to luminance (dark for shortest to bright for longest).
- *Genre* mapped to light brown flags wrapped around the glyph. Since a movie can be classified into multiple genres, multiple genre flags may appear on a glyph.



```
Sr No: 1
Title: Godfather, The Part II
Genre: Action, Crime, Drama
Year: 1974
Length: 200
Rating: 80.10
ClusterNo: 128
Location: 0.02, 1.99
```

Figure 2. A visualization of action movies generated with M_{ML} with "The Godfather, Part II" selected: the blue color of the glyph represents the movie's release date of 1974, while its higher luminance represents its length of 200 minutes; light brown flags mark the movie as *Action* and *Drama*; finally, the glyph's height reinforces its higher predicted *user rating*

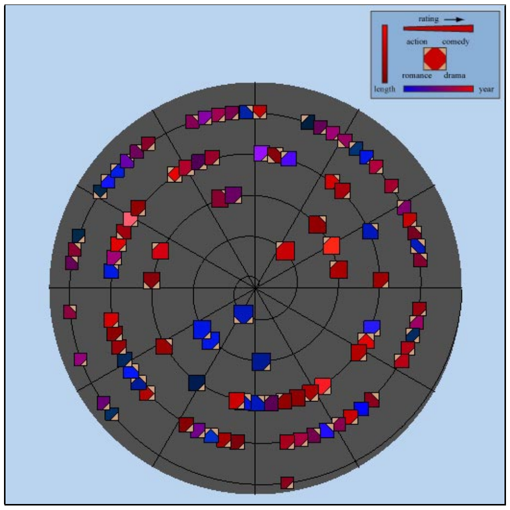
Predicted *user rating* was also used as the ranking attribute to position glyphs along the spiral. High predicted *user rating* placed a glyph near the center of the spiral, while low predicted *user rating* placed it along the outer periphery. A high spatial density of glyphs represents a cluster of results with similar predicted *user rating*. Fig. 1 shows an example of two visualizations generated by our system using M_{ML}. Fig. 2 shows a close-up of a query result containing action movies, with the glyph for "The Godfather, Part II" currently selected.

Mapping predicted *user rating* to both spatial position and size (or height) was done to emphasize the ranking attribute. Less important attributes were assigned less salient visual features. For example, luminance variations were not as easy to identify as differences in size, so they were attached to *length*, a secondary data attribute. Although both position and size represent predicted *user rating*, their scaling factors are slightly different. Position is normalized to ensure the glyphs from the query results being visualized cover the length of the spiral (Eq. 2). Size is scaled to the limits of predicted *user rating*, 1 and 100, and is independent of $A_{r,\min}$ and $A_{r,\max}$. It can therefore be used to judge the range of predicted *user rating* values in a visualization. For example, if all the glyphs have a similar size (or height), then the predicted *user rating* values in the visualization are relatively uniform.

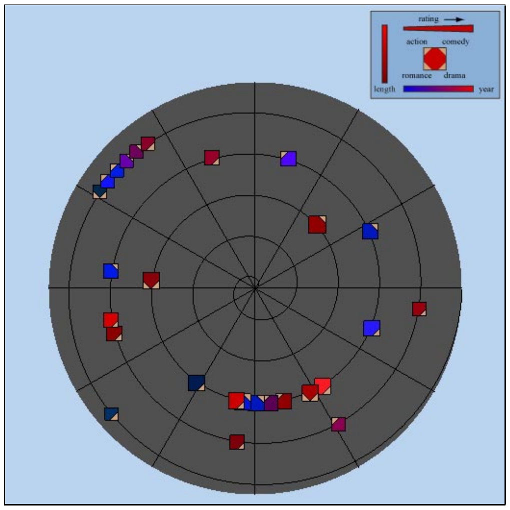
6.3. Comparing MovieLens Recommendations

Once static visualizations are constructed for results from each query, relationships between pairs of query results Q_1 and Q_2 can be displayed. We use a set of animation steps to perform this visualization, specifically:

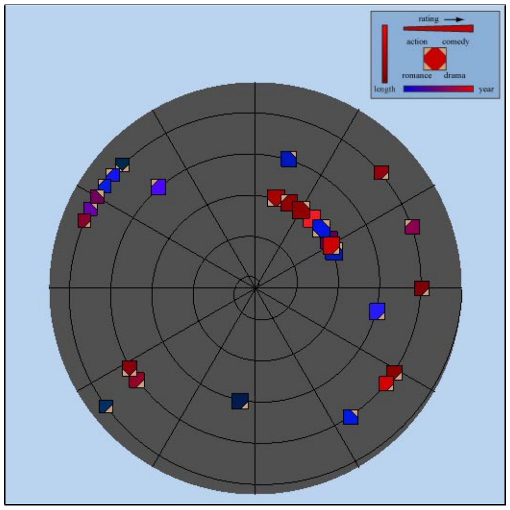
[*]www.imdb.com



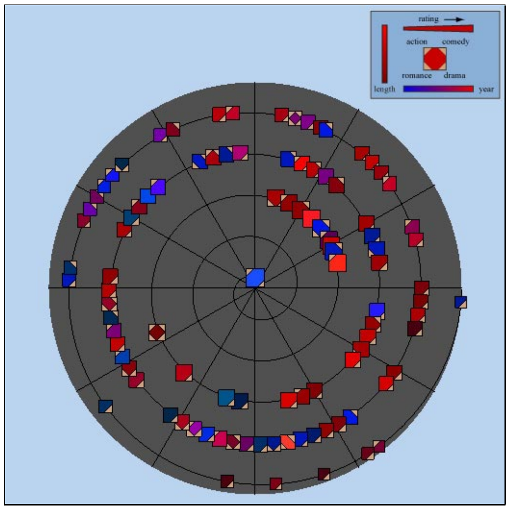
(a)



(b)



(c)

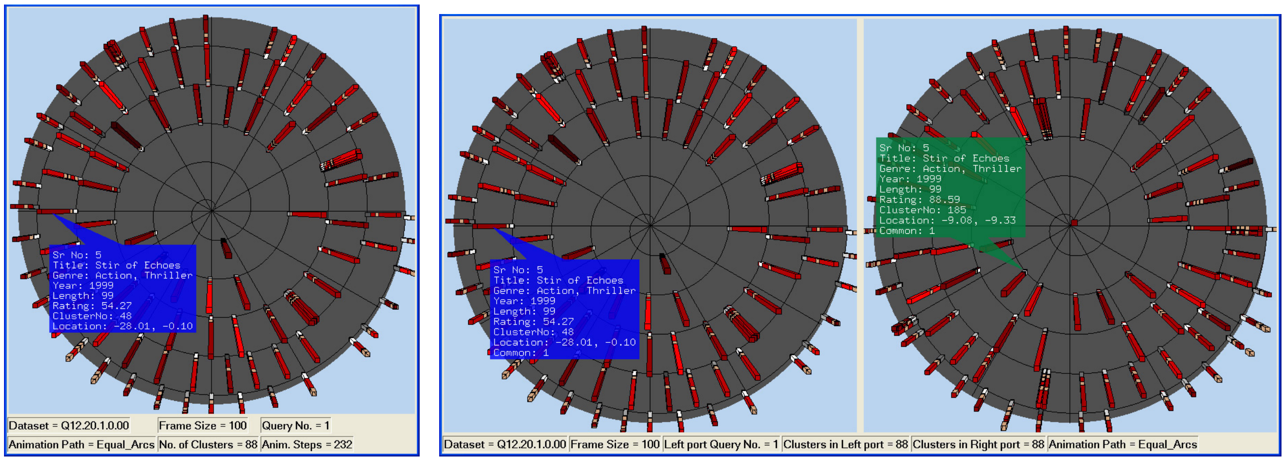


(d)

Figure 3. Visualizing query results: (a) the first friend's query results Q_1 (step 2); (b) common elements with position based on Q_1 (step 3); (c) common elements with position based on Q_2 (end of step 4); the second friend's query results (step 5)

1. Groups of movies with similar rankings are identified in both Q_1 and Q_2 using our spatial clustering algorithm.
2. Q_1 is visualized.
3. Movies unique to Q_1 are removed.
4. Movies common to Q_1 and Q_2 are moved from their current positions in Q_1 to their new positions in Q_2. Clusters containing multiple movies are moved first, followed by singletons. Movies closer to the center of the spiral move before movies on the periphery.
5. Movies unique to Q_2 are added, producing the final visualization of all the elements in Q_2.

Differences between Q_1 and Q_2 are shown in steps 3 and 5, as elements unique to Q_1 are removed, and as elements unique to Q_2 are added. Similarities between Q_1 and Q_2 are shown in step 4. Clusters of movies that maintain their positions


Sr No: 5
Title: Stir of Echoes
Genre: Action, Thriller
Year: 1999
Length: 99
Rating: 54.27
ClusterNo: 48
Location: -28.01, -0.10

Dataset = Q12.20.1.0.00 | Frame Size = 100 | Query No. = 1
Animation Path = Equal_Arcs | No. of Clusters = 88 | Anim. Steps = 232

Sr No: 5
Title: Stir of Echoes
Genre: Action, Thriller
Year: 1999
Length: 99
Rating: 54.27
ClusterNo: 48
Location: -28.01, -0.10
Common: 1

Sr No: 5
Title: Stir of Echoes
Genre: Action, Thriller
Year: 1999
Length: 99
Rating: 88.59
ClusterNo: 185
Location: -9.08, -9.33
Common: 1

Dataset = Q12.20.1.0.00 | Frame Size = 100 | Left port Query No. = 1 | Clusters in Left port = 88 | Clusters in Right port = 88 | Animation Path = Equal_Arcs

(a) (b)

Figure 4. Experiment displays: (a) the animation technique, with a single element selected (blue balloon); (b) the side-by-side technique, with a common element selected in Q_1 (blue balloon) and its position shown in Q_2 (green balloon)

relative to one another are highlighted first, followed by singletons. Starting with elements at the center of the spiral and continuing outwards to the periphery allows the viewer to track high rank (or low rank) movies from Q_1 to see how they are valued in Q_2.

6.4. Animating Query Results

Suppose two friends decided to see a movie together. Although MovieLens can produce a group recommendation of movies that both users would like, it might also be interesting for the friends to see additional information, for example, which movies one friend likes more than the other, or which movies one friend likes that the other does not like at all. Recommendations from MovieLens can be visualized to show both the intersection of movies they contain, and specific details about how the friends' movie preferences differ. Fig. 3 demonstrates the different steps in our animations. Fig. 3a represents step 2, visualizing the first friend's results Q_1. Thirty-five clusters are identified in Q_1. Position on the spiral is controlled entirely by the predicted *user rating* value. Once the initial display is built, we remove glyphs that are not present in Q_2 by gradually making them transparent (step 3). This produces a display of the intersection of the query results, with glyph positions still based on Q_1. Fig. 3b shows fifteen clusters containing a total of twenty-seven glyphs common to both Q_1 and Q_2 (i.e., twenty-seven movies neither friend has seen). In the second phase of the animation, the common glyphs move to their new positions according to the predicted *user rating* in the second friend's results Q_2 (step 4). During this phase the sizes of the glyphs increase or decrease depending on whether they are moving closer to or away from the center of the spiral. At each point during the animation phase glyphs from a single cluster are moving from their original positions to their new positions. The other glyphs remain stationary, waiting for their turn to move. The second phase ends when all the common glyphs have arrived at their new positions in Q_2 (Fig. 3c). Finally, the glyphs that are unique to Q_2 are faded in (step 5). When this step is complete, results from Q_2 are displayed in their entirety (Fig. 3d).

7. VALIDATION STUDIES

In order to further investigate our design choices, we conducted a simple validation experiment that compared our animated technique to a more traditional side-by-side display (Fig. 4). Our experiment trials used 3D glyphs to visualize data elements in each query result. Both systems allowed users to select elements to see information balloons describing their properties. This was most important for the side-by-side technique, since picking elements in one query result "brushes" the corresponding elements, if they exist, in the other result.

Subjects were asked to use each technique to answer five questions on pairs of query results. We wanted tasks that: (1) are representative of tasks users perform when comparing subsets of data elements; and (2) will help to highlight strengths and weaknesses in each technique. We chose:

- T_1: Determine the percentage of elements that are common across the two query results Q_1 and Q_2.
- T_2: Locate the highest ranked element in Q_1.
- T_3: Locate the highest ranked element in Q_2.
- T_4: Locate the element that has the highest combined ranking over Q_1 and Q_2.
- T_5: Identify a cluster of elements that maintains its local spatial neighborhood between Q_1 and Q_2.

Pairs of query results were constructed for use during the experiment. Every pair was built to contain exactly two common clusters that existed in both queries, and exactly four unique clusters that existed in only one of the queries (two unique clusters in Q_1, two in Q_2). We also varied the *percentage* of common elements (20, 40, 60, or 80%), the query result that contained the *highest* ranked element (Q_1 or Q_2), and the *difference* in predicted *user rating* between the highest elements in the two query results (2.5% or 5.0% of total user rating). Two query result pairs for each of the 16 possible conditions were built (four *percentage* by two *highest* by two *difference*), for a total of 32 pairs.

Nine faculty and graduate students from the Computer Science department (eight males and one female, all with normal or corrected vision) volunteered to participate during the experiment. Subjects were familiar with the PC computer and the Windows interface used to conduct the experiment, although none were experts in either visualization technique. Subjects performed the five tasks on all 32 query pairs, once with the animated technique, and once with the side-by-side technique. The technique each subject started with was randomized: half saw the animations first; the other half saw the side-by-sides first. Practice trials were presented before the experiment began, to explain the techniques and the tasks to the subjects, and to allow subjects to become comfortable with both techniques. The order of the trials was randomized during the experiment. Subjects completed all five tasks in order on each query pair. Subjects were told whether their answer was correct or incorrect immediately after they finished a task. Time to respond and subject accuracy were recorded for later analysis.

7.1. Results

Subject response times *rt* and error rates *e* were used to measure performance during the experiment. *rt* was recorded in seconds. *e* was zero if subjects completed a task correctly, or one if they completed it incorrectly (i.e., *e* represents percentage error). *rt* and *e* were collapsed and averaged over all subjects for each of the 32 experiment conditions: *technique* (animated or side-by-side) by *percentage* (20, 40, 60, or 80%) by *highest* (Q_1 or Q_2) by *difference* (2.5 or 5.0%). *t*-tests and analysis of variance (ANOVA) with a 95% confidence interval were used to locate significant differences in performance. Our main findings were:

1. The animation technique was significantly faster than the side-by-side technique for tasks T_1 (percentage common), T_4 (highest combined element), and T_5 (locate a cluster).
2. The animation technique had significantly fewer errors for task T_4 (highest combined element).
3. The side-by-side technique was significantly faster for task T_3 (highest in Q_2).

We compared *rt* and *e* across *technique* for all five tasks. The animation trials were significantly faster than the side-by-side trials for tasks T_1, T_4, and T_5, $t(62) = 5.382,\ p < 0.0001$, $t(62) = 12.842,\ p < 0.0001$, and $t(62) = 14.069,\ p < 0.0001$, respectively. Average response times were $\overline{rt} = 5.24$, 9.61, and 1.49 s during the animation trials, and $\overline{rt} = 6.88$, 23.51, and 6.18 s during the side-by-side trials for each of the three tasks. *e* was also significantly lower during the animation trials for task T_4, $t(62) = 2.312,\ p = 0.024$, with $\overline{e} = 0.11$ during the animation trials and $\overline{e} = 0.26$ during the side-by-side trials. The side-by-side trials were faster than the animation trials for task T_3, $t(62) = 6.46,\ p < 0.0001$, with $\overline{rt} = 2.41$ during the animation trials and $\overline{rt} = 1.92$ during the side-by-side trials.

A number of other small but significant differences were found. For task T_4 during animation trials, *rt* varied significantly over *percentage*, $F(3,28) = 3.46,\ p = 0.03$ ($\overline{rt} = 8.19$, 10.36, 10.93, and 8.94 s for 20, 40, 60, and 80% common, respectively). For task T_3 during animation trials, *rt* varied significantly over *difference*, $t(30) = 2.226,\ p = 0.034$ ($\overline{rt} = 2.53$ and 2.29 s for Q_1 and Q_2, respectively). Finally, *rt* varied significantly over *percentage* for tasks T_1 and T_5 during side-by-side trials, $F(3,28) = 6.582,\ p = 0.002$ and $F(3,28) = 4.036,\ p = 0.017$, respectively ($\overline{rt} = 5.569$, 6.361, 8.236, and 7.347 s for task T_1 and $\overline{rt} = 6.069$, 4.847, 6.097, and 7.708 s for task T_5 for 20, 40, 60, and 80% common, respectively).

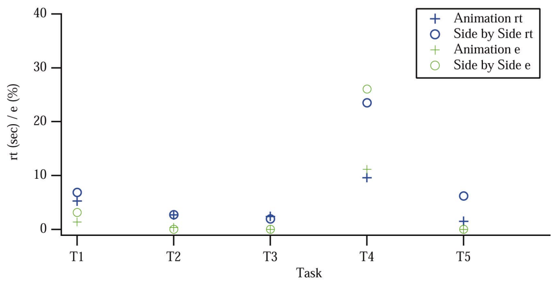


Figure 5. Graph of $\overline{rt}$ and $\overline{e}$ for the animation and side-by-side techniques, for each task

Interpretation. Animations were statistically faster and more accurate for identifying the element with the highest combined *user rating* across Q_1 and Q_2. Users "scrubbed" back and forth through the animation to find these types of elements. In the side-by-side technique users were forced to select individual elements in one query result, then look for their position in the other result. The animated technique was also faster for estimating the percentage of common elements, and for identifying clusters of elements that maintain their spatial neighborhood across both query results.

The side-by-side technique was faster for finding the two elements in Q_1 and Q_2 with the highest *user rating*, since both results were visible simultaneously. Users had to scroll back and forth in the animation technique to change the query result being displayed.

We watched to try to see how subjects completed the tasks and why they made errors. Only task T_4, highest combined *user rating*, had a large error rate (approximately 18.5%, Fig. 5). We noticed that the farther the target element moved between the query results, the harder it was to identify. Subjects also tended to "give up" in the side-by-side technique after they spent a certain amount of time searching for the target element. After completing the experiment, subjects voiced a strong preference for the animation technique.

8. CONCLUSIONS AND FUTURE WORK

Our goal is a method to visualize patterns both within and between queries on an underlying dataset. Query results are represented with perceptual glyphs positioned along a space-filling spiral, with color and texture properties used to encode the element's attribute values. Animations are used to highlight similarities and differences between pairs of query results. We used queries from a movie recommender system as a practical testbed for our visualization system. Experimental results from an initial validation study confirmed many of our anecdotal observations about the capabilities of our visualization technique.

Our technique is not restricted to movie data. For example, we are currently collaborating with bioinformatics researchers to visualize exons and associated GO terms across different transcripts from a genome database. We are also investigating how to increase the total number of data elements we can visualize at one time. For example, Fig. 6 shows 500 data elements along the spiral, which is approaching the limit for maintaining distinguishable clusters of data elements.

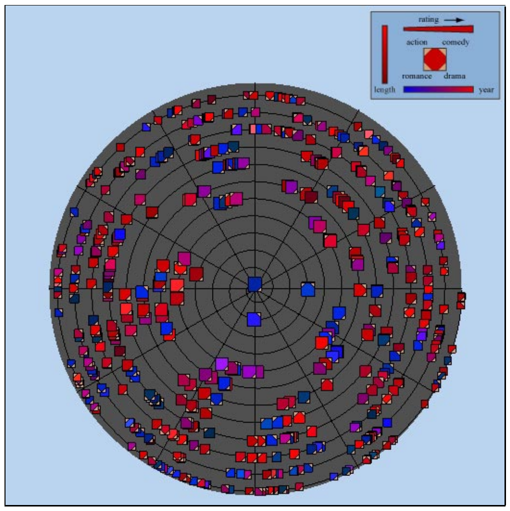


Figure 6. A MovieLens query with 500 recommendations

REFERENCES

1. J. B. Schafer, J. Konstan, and J. Riedl, "Recommender systems in e-commerce," in *Proceedings 1st ACM Conference on Electronic Commerce (EC-99)*, pp. 158–166, (Denver, Colorado), 1999.
2. S. Card, J. Mackinlay, and B. Shneiderman, *Reading In Information Visualization: Using Vision To Think*, Morgan Kaufmann Publishers, San Francisco, California, 1999.
3. M. Foltz and R. Davis, "Query by attention: Visually searchable information maps," in *Proceedings 5th International Conference on Information Visualisation*, pp. 85–96, (London, England), 2001.
4. D. A. Keim, "Pixel-oriented database visualizations," *ACM SIGMOD Record* **25**(4), pp. 35–39, 1996.

5. J. V. Carlis and J. Konstan, "Interactive visualization of serial periodic data," in *Proceedings 11th ACM Symposium on User Interface Software and Technology*, pp. 29–38, (San Francisco, California), 1998.
6. M. O. Ward and B. N. Lipchak, "A visualization tool for exploratory analysis of cyclic multivariate data," *Metrika* **51**(1), pp. 27–38, 2000.
7. M. Weber, M. Alexa, and W. Muller, "Visualizing time-series on spirals," in *Proceedings IEEE Symposium on Information Visualization (InfoVis 2001)*, pp. 7–14, (Washington, D.C.), 2001.
8. E. H. Chi, J. Pitkow, J. Mackinlay, P. Pirolli, R. Gossweiler, and S. K. Card, "Visualizing the evolution of web ecologies," in *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 400–407, ACM Press/Addison-Wesley Publishing Co., (New York, NY, USA), 1998.
9. J. Stasko and E. Zhang, "Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations," in *INFOVIS '00: Proceedings of the IEEE Symposium on Information Vizualization 2000*, p. 57, IEEE Computer Society, (Washington, DC, USA), 2000.
10. A. Spoerri, "Visualizing meta search results: Evaluating the metacrystal toolset," in *Proceedings 69th Annual Meeting of the American Society for Information Science and Technology (ASIST)*, (Austin, TX, USA), 2006.
11. A. J. Zschimmer and J. C. Roberts, "Bifocal Radial Visualization of Intranet Search Results using Image Caching," in *Proceedings of the fifth Theory and Practice of Computer Graphics*, D. Duce and I. S. Lim, eds., pp. 115–122, Eurographics, June 2007.
12. L. A. Treinish, "Task-specific visualization design," *IEEE Computer Graphics & Applications* **19**(5), pp. 72–77, 1999.
13. C. G. Healey and J. T. Enns, "Large datasets at a glance: Combining textures and colors in scientific visualization," *IEEE Transactions on Visualization and Computer Graphics* **5**(2), pp. 145–167, 1999.
14. C. Weigle, W. G. Emigh, G. Liu, R. M. Taylor, J. T. Enns, and C. G. Healey, "Oriented texture slivers: A technique for local value estimation of multiple scalar fields," in *Proceedings Graphics Interface 2000*, pp. 163–170, (Montréal, Canada), 2000.
15. A. J. van Doorn and J. J. Koenderink, "Spatial properties of the visual detectability of moving spatial white noise," *Experimental Brian Research* **45**, pp. 189–195, 1982.
16. L. Bartram, C. Ware, and T. Calvert, "Moticons: Detection, distraction, and task," *International Journal of Human-Computer Studies* **58**(5), pp. 515–545, 2003.
17. C. Ware, *Information Visualization: Perception for Design*, Morgan Kaufmann Publishers, San Francisco, California, 2000.
18. K.-P. Yee, D. Fisher, R. Dhamija, and M. Hearst, "Animated exploration of graphs with radial layout," in *Proceedings IEEE Symposium on Information Visualization (InfoVis 2001)*, pp. 43–50, (Washington, D.C.), 2001.
19. G. Robertson, K. Cameron, M. Czerwinski, and D. Robbins, "Animated visualization of multiple intersecting hierarchies," *Information Visualization* **1**(1), pp. 50–65, 2002.
20. S. Krasser, G. Conti, J. Grizzard, J. Gribschaw, and H. Owen, "Real-time and forensic network data analysis using animated and coordinated visualization," in *Proceedings from the sixth IEEE Systems, Man and Cybernetics Information Assurance Workshop*, pp. 42–49, June 2005.
21. C. G. Healey, R. St. Amant, and J. Chang, "Assisted visualization of e-commerce auction agents," in *Proceedings Graphics Interface 2001*, pp. 201–208, (Ottawa, Canada), 2001.
22. A. Sanchez and R. Orosco, "Applying animation constraints to 3D information visualization techniques," in *Proceedings Congresso Brasileiro de Computacao (CBComp 2001)*, (Santa Catarina, Brazil), 2001.
23. C. Hurley and A. Buja, "Analyzing high-dimensional data with motion graphics," *SIAM Journal on Scientific and Statistical Computing* **11**(6), pp. 1193–1211, 1990.
24. D. J. Aks and J. T. Enns, "Visual search for size is influenced by a background texture gradient," *Journal of Experimental Psychology: Human Perception & Performance* **22**(6), pp. 1467–1481, 1996.