

Visualization and Recommendation of Large Image Collections toward Effective Sensemaking

Yi Gu^a, Chaoli Wang^a, Jun Ma^b, Robert J. Nemiroff^c, David L. Kao^d, and Denis Parra^e

ABSTRACT

In our daily lives, images are among the most commonly found data which we need to handle. We present iGraph, a graph-based approach for visual analytics of large image collections and their associated text information. Given such a collection, we compute the similarity between images, the distance between texts, and the connection between image and text to construct iGraph, a compound graph representation which encodes the underlying relationships among these images and texts. To enable effective visual navigation and comprehension of iGraph with tens of thousands of nodes and hundreds of millions of edges, we present a progressive solution that offers collection overview, node comparison, and visual recommendation. Our solution not only allows users to explore the entire collection with representative images and keywords, but also supports detailed comparison for understanding and intuitive guidance for navigation. The visual exploration of iGraph is further enhanced with the implementation of bubble sets to highlight group memberships of nodes, suggestion of abnormal keywords or time periods based on text outlier detection, and comparison of four different recommendation solutions. For performance speedup, multiple GPUs and CPUs are utilized for processing and visualization in parallel. We experiment with two image collections and leverage a cluster driving a display wall of nearly 50 million pixels. We show the effectiveness of our approach by demonstrating experimental results and conducting a user study.

Keywords: large image collection, graph layout, progressive drawing, node comparison, visual recommendation

^aDepartment of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556

^bDepartment of Computer Science, Michigan Technological University, Houghton, MI 49931

^cDepartment of Physics, Michigan Technological University, Houghton, MI 49931

^dNASA Ames Research Center, Moffett Field, CA 94035

^eDepartamento Ciencia de la Computación, Pontificia Universidad Católica de Chile, Vicuña Mackenna 4860, Macul, Santiago, Chile 7820436

Corresponding author:

Chaoli Wang, Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, United States

Email: chaoli.wang@nd.edu, Telephone: 1 574 631 9212

1. INTRODUCTION

With the booming of digital cameras, image archiving and photo sharing websites, browsing and searching through large online image collections has become a notable trend. Consequently, viewing images separately as individuals is no longer enough. In many cases, we now need the capability to explore these images together as collections to enable effective understanding of large image data. Another notable trend is that images are now often tagged with names, keywords, hyperlinks and so on. Therefore, solutions that can nicely integrate images and texts together to improve collective visual comprehension by users are highly desirable.

In this work, we develop iGraph, a visual representation and interaction framework to address the increasing needs of browsing and understanding large image collections and their associated text information. These needs include the following. First, when relationships among images and texts are extracted and built in the general form of a graph, effective navigation through such a large graph representation becomes critically important. A good solution must allow collection overview and detailed exploration. This demands a flexible graph layout that dynamically and smoothly displays the relevant information content at various levels of detail. Second, visual guidance should be given so that users can easily explore the collection with meaningful directions. Besides interactive filtering, the capability to compare nodes of interest for deep comprehension is necessary. Third, automatic recommendation that provides the suggestions for further exploration is also desirable. Such a capability allows users to browse through the graph in a progressive manner.

We design iGraph to serve two kinds of users. First, for those users who are interested in but are not familiar with the given collection, iGraph not only shows the overview by introducing the backbone graph, but also lists all the images and keywords for users to go through. The visual recommendation function is also provided to help users explore the collection. Second, for the users who have some background knowledge about the collection, we design the node comparison function to allow users to study the similarities and differences among selected images and keywords. We also provide text outlier detection to allow users to easily spot anomalous nodes in the graph.

We experiment with two well-known collections: the APOD collection and the MIR Flickr collection. The Astronomy Picture of the Day (APOD)¹ is an online astronomy image collection maintained by NASA and Michigan Technological University. Everyday APOD features a picture of our universe, along with a brief explanation written by a professional astronomer. Since its debut in June 1995, APOD has archived thousands of handpicked pictures, which makes it the largest collection of annotated astronomy images on the Internet. The MIR Flickr collection² is offered by the LIACS Medialab at Leiden University. The collection was introduced by the ACM MIR Committee in 2008 as an ACM sponsored image retrieval evaluation. We use the MIRFLICKR-25000 collection which consists of 25,000 annotated images downloaded from the social photography site Flickr

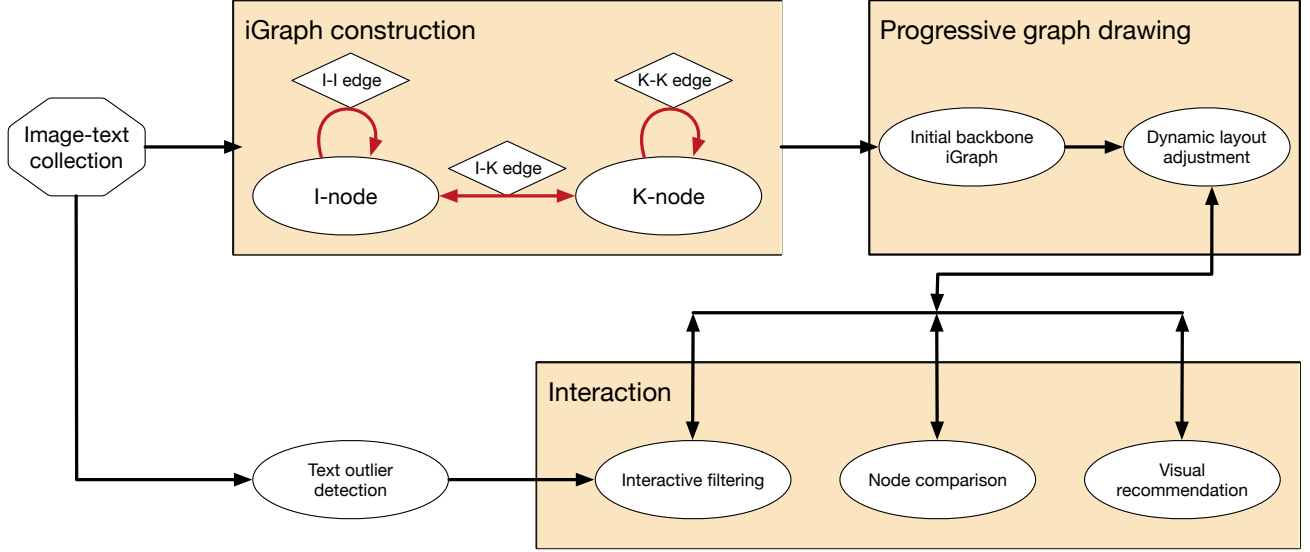


Figure 1. The overview of our approach. We construct iGraph from an image and text collection. Progressive drawing is introduced to draw the iGraph during user navigation. We provide a set of interaction functions to help users explore the entire collection.

through its public API.

As sketched in Figure 1, we extract image and text information from each collection, analyze their similarities to build interconnections, and map extracted data and their relationships to a new compound graph representation. Our iGraph consists of tens of thousands of nodes and hundreds of millions of edges. To enable effective exploration, we incorporate progressive graph drawing in conjunction with animated transition and interactive filtering. Rather than building a fixed hierarchical organization of images and texts, our notion of *progressive graph drawing and adjustment* allows more flexible and desirable navigation. Node comparison is enabled by optimally arranging selected nodes and their most related ones for detailed analysis and display. We provide various means for image and keyword input so that users can conveniently select nodes of interest for comparisons. To provide effective guidance, automatic visual recommendation is realized by providing the suggestions for future exploration based on the analysis of image popularity, text frequency, and user exploration history. To complement the overview of iGraph, we also perform text outlier detection to automatically suggest abnormal keywords or time periods in the graph for exploration. We leverage the bubble set visualization to clearly highlight results for node comparison and visual recommendation. We implement four different recommendation solutions and compare their recommendation quality. The performance of iGraph is enhanced with the utilization of multiple GPUs and CPUs in the processing and visualization, along with a display wall consisting of 24 monitors. We also conduct a user study to evaluate the effectiveness of iGraph.

2. RELATED WORK

2.1 Visualization of Image and Text Collections

Visualizing Image Collections. Organizing image collections into various visual forms has been a well-studied topic in the graphics, imaging, and visualization community. Chen et al.³ organized the images using the Pathfinder networks based on image colors, textures, and shapes. Bederson⁴ designed PhotoMesa which arranges images using quantum treemaps and bubblemaps for zoomable image browsing. Platt et al.⁵ developed PhotoTOC which uses a temporally ordered list of all photographs from a user as the detailed view and utilizes an automatic clustering algorithm to generate the overview for personal photograph browsing. Torres et al.⁶ introduced spiral and concentric rings for focus+context visualization in conjunction with content-based image retrieval. Jankun-Kelly and Ma⁷ presented MoireGraphs which leverages a focus+context radial graph to layout nodes and their associated images. MoireGraphs also provides several interaction techniques, such as focus strength changing, radial rotation, level highlighting, secondary foci, and animated transition, to assist graph exploration. Yang et al.⁸ proposed the semantic image browser (SIB) that organizes images using an image layout based on multidimensional scaling. SIB includes the value and relation display (which allows effective high-dimensional visualization without dimension reduction) and several interaction tools (e.g., searching by sample images and content relationship detection). Brivio et al.⁹ presented dynamic image browsing which partitions the screen with weighted Voronoi diagrams to visualize images of different sizes, orientations, and aspect ratios. Wang et al.¹⁰ designed iMap, a treemap-based representation for visualizing and navigating image clustering and search results. Their layout places the query image at the center of the display area and arranges other images based on image similarity along a spiral shape from inside out. Compared with the iGraph representation,¹¹ iMap makes much more effective use of the available display area in the layout. Zhang et al.¹² studied the effectiveness of animated transitions in a tiled image layout by comparing different animation schemes. They concluded that users completed the tasks faster and more accurately using multi-step animated transition compared with no-animation and one-step animation solutions. Therefore, we leverage multi-step animated transition (fade-out, fade-in, and node movement) to help users follow the transitions in iGraph.

Visualizing Text Collections. Besides images, text analytics and visualization has received lots of attention recently. Several survey papers^{13–17} discuss text visualization and its related research problems and challenges. There are notable text visualization examples. For instance, Wattenberg and Viegas¹⁸ designed the word tree, a graphical version of the traditional “key word in context” method which allows users to explore the text body, find interesting words, and view additional phrases. Clarkson et al.¹⁹ presented ResultMaps which utilizes the treemap representation to enhance query string-driven digital library search engines. It maps each repository document into a treemap and highlights query results. ResultMaps avoids users from misled by the rank of the

search results, provides better preview, and makes useful connections between documents.

Word cloud, also known as tag cloud, is a widely used visual representation for text visualization. Words are usually single words and their importance is mapped to font sizes or colors. Straightforward word clouds organize words as lines by lines. This allows quick understanding of the popular or important words. Since different font sizes are used to arrange words lines by lines, there are gaps between lines. In addition, their typefaces are usually limited to the standard browser fonts. Wordle²⁰ was designed to address these problems. The Wordle layout algorithm allows word rotation in order to reduce gaps between words. Plenty of typefaces and color themes are included to enrich the visual experience of Wordle. ManiWordle²¹ further allows users to adjust the typefaces, color themes, and word angles, not only for the layout as a whole, but also individual words. Context preserving dynamic word cloud²² utilizes a force model to adjust word placement in order to reduce and even avoid overlap. Originated from word cloud, tree cloud²³ utilizes a tree structure to show the semantic proximity of the words. SparkClouds²⁴ combines sparklines with word cloud where a sparkline is shown right beneath each word to show its trend. Newdle²⁵ visualizes large online news collections using Wordles. It first constructs two networks (an article network and a bipartite article and tag network) and utilizes the networks as a background structure. The clusters in the network represent different topics. To visualize the tags in each topic, Newdle shows the Wordles of the largest clusters as the overview. The detailed view not only shows the Wordles, but also lists their related articles.

ThemeRiver²⁶ utilizes a river metaphor to convey the evolution of thematic contents. In the 2D plot, the horizontal direction represents the time and the vertical direction represents the strength of selected topics. Different colors are used to represent different topics. Built on the ThemeRiver metaphor, TIARA²⁷ is a text analytic system that shows the content evolution of multiple topics. Users are allowed to expand each topic to study its underlying keywords and evolution over time.

Similar to the text visualization systems (e.g., Wordle) that focus on the importance of text, iGraph also presents the most important keywords in its backbone layout. In addition, it uses colors to encode the importance of keywords. While ThemeRiver and other systems visualize the evolution of text, iGraph detects the outliers among keywords for user query. Perhaps the most important difference between iGraph and other systems is that iGraph focuses on the relations not only between keywords, but also between images and keywords, and between images. By presenting the related images and keywords, iGraph allows users to get a direct and strong impression of what the selected image or keyword means. This benefit, however, can not be achieved by visualizing images and texts separately.

Visualizing Image and Text Collections. Despite the abundant research on visualizing images and on visualizing texts, considering both images and texts in the same visualization has not been thoroughly

investigated. Several techniques tried to reorganize texts and images of a document to fit into small screen display, such as paper-to-PDA²⁸ and SmartNails.²⁹ Document Card,³⁰ on the other hand, visualizes a document with the extracted keywords and images. It adopts the idea of trumps game cards to provide a combined overview of an object. Each card consists of images and associated keywords which describe a document. Different from Document Card which utilizes the images and keywords to explain the entire document, iGraph explains each image and keyword using its related images and keywords. For a large image and text collection, making connection of all related items and studying the relations between them become very important. This paper aims to address this issue. Our iGraph analyzes the similarities between images and images, keywords and keywords, images and keywords. We visualize the relations between not only the same, but also different kinds of nodes in a single visualization. Although iGraph is implemented on the image collections associated with keywords, it can be applied to the visualization of general image and text collections.

2.2 Graph Drawing

Graph drawing is an important area of research in visualization. There are well-known algorithms, e.g., those developed by Fruchterman-Reingold,³¹ Kamada-Kawai,³² Davidson-Harel.³³ There are also some generalized expectation-maximization algorithms. Several survey papers discussed graph layout methods and performance comparisons.^{34–37} Recent trends on graph drawing are on large-scale complex graphs, such as dynamic, compound, and online graphs. One of the most important problems for graph drawing is how to handle the ever-growing size of graphs. Walshaw³⁸ introduced a multilevel force-directed layout algorithm which groups nodes into clusters and renders the graph in a coarse-to-fine manner. Holten and van Wijk,³⁹ on the other hand, utilized the edge bundling technique to reduce visual clutter and highlight edge patterns. Zinsmaier et al.⁴⁰ focused on dense edges and proposed a technique that allows straight-line graph drawings to be rendered interactively with adjustable levels of detail. Researchers have also investigated GPU/CPU-accelerated solutions for drawing large-scale graphs and drawing graphs on large displays. For example, Frishman and Tal⁴¹ presented a multi-level graph layout on the GPU. Bastian et al.⁴² developed Gephi, an open source software that utilizes the 3D engine to display large graphs. Gephi also supports dynamic graph drawing through dynamic features presentation. Alper et al.⁴³ presented the stereoscopic highlighting technique for 2D graph visualization and ran their experiment in the Allosphere Virtual Reality environment featuring a large spherical display with a 5m radius. iGraph utilizes the modified Fruchterman-Reingold force-directed layout algorithm for graph drawing. Since the forces are independent from each other, we can easily parallelize the algorithm to tackle the ever-growing sizes of graphs produced.

Another important problem is the removal of node overlaps since nodes usually carry sizes and shapes. Some researchers utilized the post-processing approach. They first placed nodes in the graph, then removed/reduced

the overlaps. Dwyer et al.⁴⁴ introduced a fast node overlap removal algorithm for 2D graph drawing. They first removed the overlaps in one dimension and then introduced restrictions to remove the overlaps in the other dimension, which transfers the node removal problem to a quadratic programming problem. Gansner and Hu⁴⁵ first approximated a graph based on the original graph. Using this graph as the guidance, they iteratively moved the nodes, especially the overlapping nodes. At the same time, they kept the relative positions between nodes as close as possible to those in the original layout. Other researchers considered node sizes in the node removal problem. For instance, based on the Kamada-Kawai layout,³² Dwyer et al.⁴⁶ presented IPSEP-COLA, an algorithm that leverages stress majorization to minimize a series of quadratic forms with guarantee of stress decrease. Their incremental algorithm is based on gradient projection for the quadratic solving problem as well as the predefined separation constraints. To handle this problem, iGraph utilizes *constrained layout adjustment*, *density adjustment*, *K-node overlapping adjustment*, and *occluded I-node removal*.

Drawing Dynamic Graphs. Graph updating is an operation that inserts or removes nodes or edges iteratively. Drawing a dynamic or time-varying graph consists of a serial of such operations. Greilich et al.⁴⁷ developed TimeArcTrees which visualizes weighted, dynamic compound digraphs by drawing a sequence of node-link diagrams in a single view. Burch et al.⁴⁸ presented TimeSpiderTrees based on a radial layout. Starting from a node-link diagram, TimeSpiderTrees first gets its half-link version, and then places the nodes along a circle. After that, nodes at the same time step are placed in the same layer to form a radial layout. For a compound graph, additional information is finally rendered at the outer-most ring. Yi et al.⁴⁹ designed TimeMatrix, a matrix-based graph visualization to support dynamic graph analysis. Handlak et al.⁵⁰ introduced the concept of in situ visualization that tightly integrates existing visualization techniques for visualizing large dynamic networks. Archambault et al.⁵¹ studied which approach works best for reading dynamic graphs and whether preserving the mental map helps graph reading or not. Feng et al.⁵² presented an algorithm for coherent time-varying graph drawing, supporting the capability of multi-focus+context interaction and visualization. Bach et al.⁵³ presented GraphDiaries, which renders the dynamic graph in animation and highlights the changes of graph in the adjacent time steps to help the understanding. Beck et al.⁵⁴ categorized dynamic graphs into two representations: animated diagrams or timeline-based static charts. To draw the dynamic graph, iGraph utilizes animated transition to preserve the mental map.

Drawing Compound Graphs. A compound graph is a node-link diagram with multiple types of nodes or links. Singh et al.⁵⁵ developed Invenio, a visual mining tool for multi-model social network data mining. Invenio allows users to interactively explore the networks through creating views that support visual analytics using both database operations and basic graph mining operations. Burch et al.⁴⁸ designed TimeSpiderTrees which uses a radial layout representation to visualize a dynamic compound graph. Ghani et al.⁵⁶ presented MMGraph, a visual analysis tool to visualize multi-model social networks. In this interactive tool, different colors are used to

distinguish different models. MMGraph also utilizes parallel node-link bands to reduce visual clutter. iGraph integrates images and texts together in a compound graph that consists of two kinds of nodes (I-nodes and K-nodes) and three kinds of edges (I-I edges, I-K edges, and K-K edges).

2.3 Our Contributions

Our iGraph strives for flexible and desirable navigation as images and texts are not organized hierarchically. Rather than focusing on images alone, by integrating images and texts together in a compound graph, we allow users to explore the entire collection by navigating through images and texts of interest. Dynamic visualization of the large iGraph is achieved by progressive drawing in conjunction with interactive filtering. Node comparison and visual recommendation are enabled for user-guided detail comprehension and graph exploration. We leverage a cluster to speed up both preprocessing and runtime visualization. We also demonstrate the performance of iGraph by delivering the visualization results to a large display wall.

3. IGRAPH DEFINITION AND CONSTRUCTION

We define iGraph as a compound graph that consists of two kinds of nodes, i.e., I-node and K-node, and three kinds of edges, i.e., I-I edge, I-K edge, and K-K edge, where “I” stands for image and “K” stands for keyword. An I-node (K-node) represents an image (keyword) extracted from the collection. An I-I edge (K-K edge) is formed between two I-nodes (K-nodes) and the edge weight indicates the similarity between these two I-nodes (K-nodes). Finally, an I-K edge is formed between an I-node and a K-node only if there is a connection between them.

3.1 I-node and K-node

In iGraph, each I-node corresponds to an image. Each K-node corresponds to a keyword related to more than one image (otherwise, we treat it as a rare keyword and exclude it from iGraph). Keywords are given as tags in the MIR Flickr collection. For the APOD collection, we extract meta-tagged keywords from the HTML header and other keywords from the paragraph of explanation accompanying each image.

In the explanation sections, we first extract the nouns using a tag extraction tool named “Stanford Log-linear Part-Of-Speech Tagger”.⁵⁷ Then we combine the keywords extracted from the HTML header and explanation sections. Finally, we manually check the keywords to eliminate the low-frequent or meaningless keywords. We use frequencies of the keywords as their importance instead of “term frequency-inverse document frequency” (TF-IDF) because of the low reappearance of the keywords in the same webpage. TF-IDF considers the overall frequencies and the number of appearing webpages of each keyword. In the APOD collection, since the explanation sections are short, the keywords barely appear twice. Therefore, the frequencies of the keywords are

approximately the same as the number of webpages which they appear in. We thus use the frequencies as their importance.

3.2 I-I edge, K-K edge, and I-K edge

An I-I edge exists between any two I-nodes and the edge weight is the similarity between the two images. Different images in the collection come with different dimensions, types, and formats. For simplicity, we convert all images to the same type and format (i.e., portable pixmap format, PPM), and scale them down to a fixed resolution (256×256) for similarity analysis. Such a resolution strikes a good balance between maintaining image content and achieving comparison efficiency. We consider three aspects of images, namely, *grayscale content*, *power spectrum*, and *color histogram*, to calculate the distance between two images.¹⁰ The overall distance between two images is a weighted sum of the three partial distances.

An K-K edge exists between any two K-nodes and the edge weight is the similarity between the two keywords. To analyze the similarity between two keywords, Gomaa and Fahmy⁵⁸ categorized the approaches into string-based, corpus-based, knowledge-based, and hybrid similarity measures. A corpus-based similarity measure is a semantic similarity measure. It determines the similarities between keywords according to information gained from large corpora. This kind of measures usually provides better similarity measurement than string-based measures. A knowledge-based similarity measure is also a semantic similarity measure, but it uses information derived from semantic networks. Among corpus-based similarity measures, we choose Google similarity distance (GSD)⁵⁹ due to its simplicity.

Given that a keyword K_a appears in a webpage, its GSD to another keyword K_b is computed as

$$GSD(K_a, K_b) = \frac{\log \max\{f_a, f_b\} - \log c_{ab}}{\log MAX - \log \min\{f_a, f_b\}}, \quad (1)$$

where f_a and f_b are the frequencies of K_a and K_b , respectively, c_{ab} is their co-occurrence frequency, and MAX is the multiplication of the largest two frequencies in all keywords. As we can see, if K_a and K_b always appear simultaneously, their distance is zero. When the frequencies of K_a and K_b are constant, their distance gets smaller if they share more webpages. When the number of sharing webpages is constant, their distance gets larger if the number of non-sharing webpages increases.

An I-K edge exists between an I-node and an K-node if and only if the image has the keyword in its tags in the MIR Flickr collection or the image and keyword exist in the same webpage of the APOD collection.

4. PROGRESSIVE DRAWING

4.1 Initial Backbone iGraph

Direct drawing the entire iGraph consisting of over tens of thousands of nodes incurs a heavy computational cost and usually produces a poor visualization, making the subsequent exploration very difficult. We therefore advocate the visual analytics mantra: “*analyze first, show the important, zoom, filter and analyze further, details on demand*”⁶⁰ in the drawing. Specifically, we first extract the backbone of iGraph by identifying representative I-nodes and important K-nodes from the graph for overview. We apply *affinity propagation*⁶¹ to the image collection to identify representative I-nodes. Unlike k-means and k-medoids clustering algorithms, affinity propagation simultaneously considers all data points as potential exemplars and automatically determines the number of clusters. Affinity propagation is a clustering algorithm based on message passing between data points. There are two kinds of message exchanges: “responsibility” and “availability”. The *responsibility* from data point p to exemplar e indicates how well e serves as the exemplar of p . The *availability* from exemplar e to data point p indicates how well for p to take e as its exemplar. The message passing procedure updates all the responsibilities and availabilities between pairs of data points, determines the exemplars by combining the responsibilities and availabilities together, and terminates when a predefined criteria is met. For K-nodes, we rank their corresponding keywords based on the frequency to determine their importance.

For an I-I edge (K-K edge), the edge weight is defined as the similarity of the two incident I-nodes (K-nodes). For any I-K edge, we define the edge weight as 1 (0) if there is a (no) connection between the I-node and the K-node. We modify a classical force-directed graph layout algorithm, the Fruchterman-Reingold (FR) algorithm,³¹ to draw the initial iGraph. Specifically, we design two forces, the repulsive force F_r and attractive force F_a . Given two nodes with their graph distance d and similarity σ , the two forces are calculated as $F_r = k_r/d$ and $F_a = k_a \times d^2 \times \sigma^2$, where k_r and k_a are the constant factors for the repulsive and attractive forces, respectively. We add σ^2 in the attractive force to ensure that more similar nodes are closer to each other. This is not included in the repulsive force, because we want to ensure that there is always a distance between any two nodes.

4.2 Dynamic Layout Adjustment

As a user explores iGraph, we dynamically adjust it around areas of interest to allow navigation through the graph in a progressive manner. The graph keeps updating as the new images or keywords of focus are selected. Note that each I-node or K-node displayed in the graph occupies a certain rectangular area. We map the size of an I-node to the importance of the corresponding image. For K-nodes, we use the same font size, while their frequency is mapped to color saturation (bright red to dark red). The most challenging issue for dynamic layout adjustment is to reduce their overlap or occlusion. In particular, we do not allow an K-node to have any overlap with other nodes, while two I-nodes could overlap each other but we want to reduce such an overlap as much

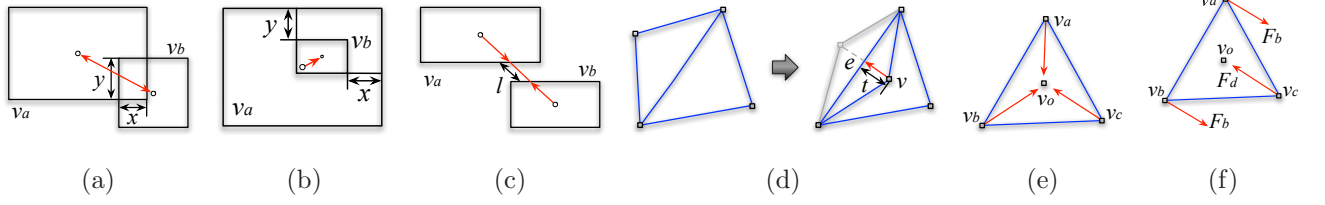


Figure 2. Four forces for constrained layout adjustment: (a) bidirectional repulsive force, (b) unidirectional repulsive force, (c) spring force, and (d) attractive force. Two forces for density adjustment: (e) density force and (f) bounding force.

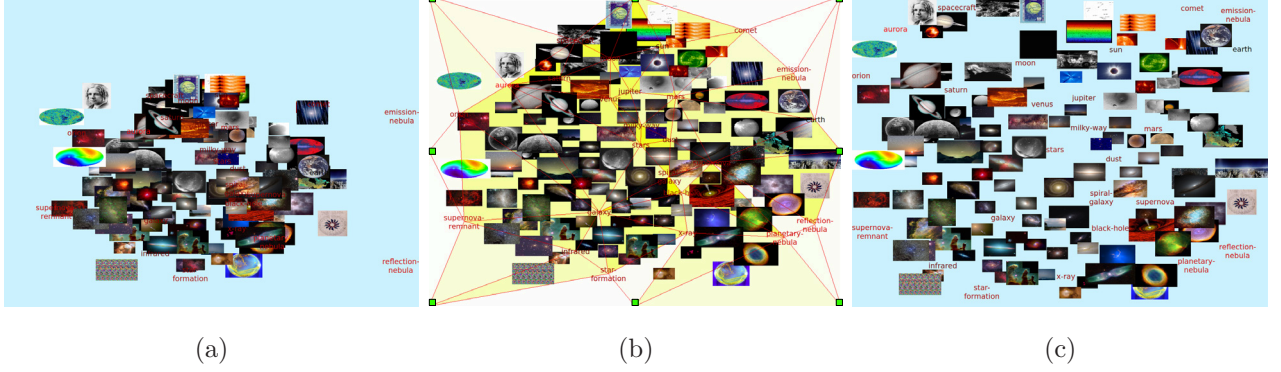


Figure 3. The process of dynamic iGraph layout adjustment. (a) The result of the modified FR algorithm. (b) The triangle mesh based on K-nodes after constrained layout adjustment. The colors of the triangles indicate their density values (higher saturation, higher density). Eight bounding nodes are marked in green. (c) The layouts after K-node overlapping adjustment and occluded I-node removal.

as possible. A good layout adjustment algorithm should also maintain a good balance between preserving the structural information of the graph and revealing the dynamics.

To this end, we generate an initial layout for the backbone iGraph. To achieve stable graph update, good screen utilization, and node overlapping reduction, we introduce four adjustment steps: *constrained layout adjustment*, *density adjustment*, *K-node overlapping adjustment*, and *occluded I-node removal*. For constrained layout adjustment, we apply a triangulation scheme⁶² to all the nodes in the initial graph and use the resulting mesh to perform the adjustment. Similar to the algorithm described by Cui et al.,²² we consider four kinds of forces to reposition the nodes to reduce their overlap while maintaining the topology of iGraph. These forces are:

- *Bidirectional repulsive force*: This force pushes away two nodes v_a and v_b from each other and is effective only if v_a and v_b overlap each other. It is defined as $F_1(v_a, v_b) = k_1 \times \min(x, y)$, where k_1 is a given weight and x and y are the width and height of the overlapping (see Figure 2 (a)).
- *Unidirectional repulsive force*: This force pushes away a node v_b from a node v_a and is effective only if v_b is inside v_a . It is defined as $F_2(v_a, v_b) = k_2 \times \min(x, y)$, where k_2 is a given weight and x and y are the

width and height of the gap region (see Figure 2 (b)).

- *Spring force*: This force balances the graph by offsetting the two repulsive forces introduced. Given two nodes v_a and v_b , the spring force is defined as $F_3(v_a, v_b) = k_3 \times l$, where k_3 is a given weight and l is the length of the line segment connecting the centers of v_a and v_b that lies outside of their boundaries (see Figure 2 (c)).
- *Attractive force*: This force maintains the topology of the triangle mesh we construct for the graph. During layout adjustment, a triangle may be flipped (see Figure 2 (d)). Our goal is to maintain stable update of the graph by introducing an attractive force to flip the triangle back. The attractive force is define as $F_4(v) = k_4 \times t$, where k_4 is a given weight and t is the distance from node v to edge e . We also consider virtual triangle edges connecting extreme nodes in the graph to the *bounding nodes* (i.e., four corners and the midpoints of four sides of the drawing area, see Figure 3 (b)). This is to ensure that all graph nodes do not go out of bound.

For density adjustment, we apply the same triangulation scheme to the K-nodes only. For each triangle in the resulting mesh, we calculate its density value ρ as $\rho = \sum_i A_i / T$, where A_i is the size of the image whose center is in the triangle and T is the size of the triangle. Then we introduce the following two forces:

- *Density force*: The density force F_d keeps the triangle area proportional to its density. As shown in Figure 2 (e), for each node, F_d pulls it to the center of the triangle. The force is calculate as $F_d = k_d \times \rho$, where k_d is a given constant.
- *Bounding force*: If we only apply the density force, the eight bounding nodes will be pulled toward to the drawing center. However, since the bounding nodes are fixed, all the rest of points will be pulled to the drawing center. To balance the effect that F_d works on the bounding nodes, we introduce the bounding force F_b . In Figure 2 (f), assume v_c is the bounding node and v_a and v_b are not. F_d on v_c is the density force, F_b on v_a and v_b is the bounding force which has the same magnitude as F_d but in the opposite direction.

For K-node overlapping adjustment, we reduce the overlapping with any K-node by adjusting the positions of the nodes which overlap with an K-node. Figure 3 shows an example of our dynamic layout adjustment results. As we can see, constrained layout adjustment nicely reduces node overlap while maintaining the graph topology. Density adjustment is able to pull nodes further apart and further reduces their overlap. Finally, K-node overlapping adjustment dictates that no K-node should overlap any other node. This is to make sure that all keywords are easily readable.

For occluded I-node removal, we calculate for each I-node, the percentage of its pixels overlapped with any other I-node. When the largest percentage of all I-nodes is larger than a given threshold, we simply remove the corresponding I-node from the graph and update the percentages for the remaining I-nodes. We repeat this removal process until the largest percentage is smaller than the threshold.

We did not apply IPSEP-COLA⁴⁶ because the FR algorithm usually works faster than the algorithms using quadratic solvers. IPSEP-COLA even allows users to predefine the minimal distances between nodes. In iGraph, although we do not want nodes overlapping with each other, due to the sizes of images and texts as well as the number of image and text items displayed, overlapping within the given display area often becomes inevitable. Therefore, the predefined minimal distance does not fit our purpose. On the other hand, our force model can reduce the overlaps and better utilize the screen space. In addition, different from the algorithms using quadratic solvers, the placement of each node in iGraph can be calculated individually. Therefore, our algorithm can be further accelerated using the parallel algorithm as described in Section 8.

4.3 Graph Transition

To preserve the mental map (i.e., the abstract structural information a user forms by looking at the graph layout) during graph exploration, we provide animated transition from one layout to another by linearly interpolating node positions over the duration of animation. Besides the compound graph, we also provide users with the option to observe the image or keyword subgraph only in a less cluttered view through animated transition. During graph transition from G_1 to G_2 , we first calculate all the node positions in G_1 and G_2 . Then we fade out the nodes that are in G_1 but not in G_2 , and fade in the nodes in G_2 but not in G_1 to form the intermediate graph G' . Finally, we move the nodes to their final locations by linearly interpolating node positions from G' to G_2 over the course of animation.

5. FILTERING, COMPARISON AND RECOMMENDATION

5.1 Interactive Filtering

We provide interactive filtering to users for sifting through the graph to quickly narrow down to images or keywords of interest. For images, users can scan through the entire list of images to identify the ones they would like to add into iGraph. They can also type a keyword to retrieve related images for selection. For keywords, users can scan through the entire list of keywords to identify the ones or type a keyword prefix to find matched ones for selection. Users can also sift through the graph according to the time information. They can not only retrieve the images and keywords in a particular time period, but also click a keyword to retrieve its related images in the time period for selection. For keywords that are already in the graph, we highlight them in red while the rest in the list are in black. For images, we draw them with lower opacity in the list if they are already

in the graph. In addition, users can also dynamically adjust the number of images or keywords they would like to display in the current view.

5.2 Node Comparison

Our node comparison allows users to compare nodes of interest for detail comprehension. Similar to the work of PivotPaths,⁶³ we allow users to select or input two to four nodes for comparison. These nodes will be moved to fixed positions around the drawing center. For each selected I-node (K-node), we retrieve n most similar I-nodes (K-nodes) and m related K-nodes (I-nodes) as its group, where n and m are user-defined parameters. Then, we remove the nodes from the previous layout which are not in any group being compared. After that, we apply the modified FR algorithm while fixing the positions of selected nodes, fade in the nodes that are not in the previous layout, and dynamically adjust the layout while the selected nodes are given the freedom to move. The reason is that the layout will be very cluttered in most cases if we always fix the positions for the selected nodes. Meanwhile, the mental map would still be preserved since the selected nodes will not change their positions dramatically compared to their previously fixed positions during the last step of the adjustment. We assign different colors to the selected nodes to indicate their group memberships.

5.3 Visual Recommendation

To allow exploring iGraph with no prior knowledge, it would be desirable for our system to show only the most relevant portions of the graph to users, while suggesting directions for potential exploration. When a node is selected, we apply collaborative filtering to recommend related nodes. These recommendations are highlighted in a focus+context manner. User interaction history is saved as the input to collaborative filtering.

Used by Amazon.com, the collaborative filtering⁶⁴ recommends items to customers based on the item they are currently viewing. We keep an *item-user matrix* \mathbf{N} where the rows are the items (images and keywords in our scenario) and the columns are the users. This item-user matrix records how many times an item has been selected by a user. Unlike the collaborative filtering which recommends the items based on all the items that the current user has selected, we only recommend based on item I_j that the user just selected. An array A is created, where each entry A_i records the number of users that selected both items I_i and I_j ($i \neq j$). Then we sort A in the decreasing order and recommend the first n items while n is a user-defined number. We refer to this solution as the *user-centric* approach.

Another solution is the *item-centric* approach. Initially, when we either do not have many users or do not have much user exploration history, we mostly make recommendations based on the similarity between items. That is, we recommend nodes that are most similar to the node selected. We provide three similarity metrics for users to select. The first metric is the direct use of the similarity matrix defined in iGraph, which we call the

primitive approach. The second and third ones are derived using the rooted PageRank⁶⁵ and Katz⁶⁶ algorithms, respectively. With an increasing number of users using iGraph leading to richer user exploration history, we are able to gradually shift from the item-centric approach to the user-centric approach and effectively increase the number of items recommended. In the following, we discuss the rooted PageRank and Katz algorithms in detail.

Given a graph, the hitting time $H_{(n_i, n_j)}$ measures the estimated number of steps from one node n_i to another node n_j using random walks. There are two issues with this measure. First, if n_j has a large stationary probability, the hitting time $H_{(n_i, n_j)}$ is very small from any given node n_i . Second, this measure is sensitive to the subgraph far away from n_i and n_j even if they are very close to each other. This is because the random walks may walk to a subgraph far away from them and cannot return to n_i and n_j within a few steps, which increases the hitting time. The rooted PageRank⁶⁵ was designed to solve these two issues. To solve the first issue, it considers the stationary probability along with the hitting time in the measurement. To solve the second one, it allows the random walks from n_i to n_j to periodically return to n_i with a probability a .

Assuming a graph is represented using an adjacency matrix \mathbf{A} , we compute the degree matrix \mathbf{D} as

$$\mathbf{D}_{i,j} = \begin{cases} \frac{1}{\sum_{k=1}^N \mathbf{A}_{i,k}}, & i = j \\ 0, & i \neq j \end{cases} \quad (2)$$

where N is the number of nodes in the graph. The probability matrix \mathbf{P} measures the probability of walking from any node to its neighbors in a single step with a uniform probability, where

$$\mathbf{P} = \mathbf{DA}. \quad (3)$$

Then we compute the similarity matrix \mathbf{S} as

$$\mathbf{S} = (1 - a)(\mathbf{I} - a\mathbf{P})^{-1}, \quad (4)$$

where a is the probability of returning to the starting node and \mathbf{I} is the identity matrix.

Katz⁶⁶ argued that the similarity between two nodes is not only related to how similar they are, but also related to how many others are similar to them. The similarity $S_{i,j}$ between two nodes n_i and n_j is computed as

$$S_{i,j} = \sum_{l=1}^{\infty} a^l \times |\mathbb{P}(n_i, n_j)^l|, \quad (5)$$

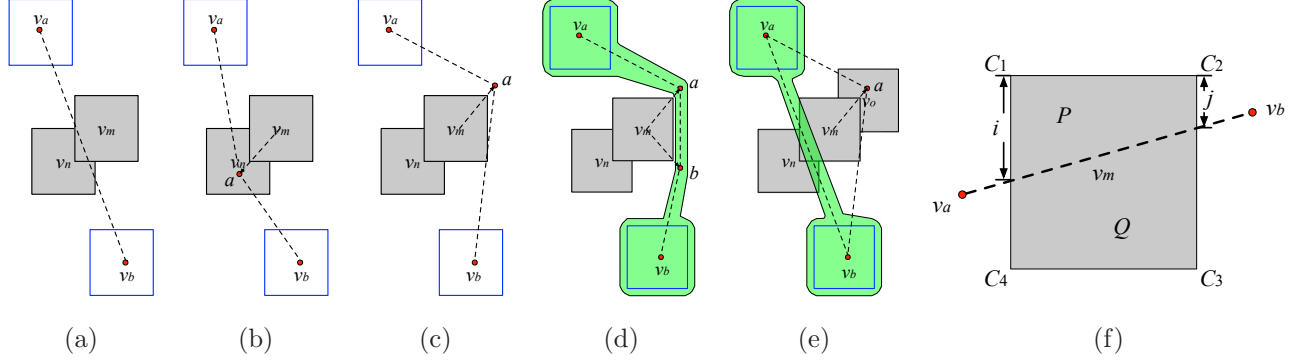


Figure 4. (a) A backbone edge $v_a v_b$ in S_b overlaps with nodes v_m and v_n in S_r . (b) The route tries to find an intermediate point a to avoid crossing v_m , but a is still in v_n . (c) The route tries the opposite corner of v_m and finds a so that $v_a a$ does not overlap with v_m . However, av_b still overlaps with v_m . (d) By adding a new point b , the final route avoids overlapping with v_m and v_n . The green region indicate the corresponding bubble set. (e) In another example, a is still in $v_o \in S_r$. Therefore, this algorithm could not always avoid the overlapping. In this case, we simply use the original backbone edge $v_a v_b$ as the route. (f) A backbone edge $v_a v_b$ in S_b overlaps with node v_m . The routing points can be identified using Algorithm 1.

where $\mathbb{P}(n_i, n_j)^l$ is the set of paths from n_i to n_j with path length l , and a is a weight between 0 and 1. Assuming a graph is represented as an adjacent matrix \mathbf{A} , the similarity matrix \mathbf{S} is computed as

$$\begin{aligned} \mathbf{S} &= a\mathbf{A} + a^2\mathbf{A}^2 + \dots + a^k\mathbf{A}^k + \dots \\ &= (\mathbf{I} - a\mathbf{A})^{-1} - \mathbf{I}. \end{aligned} \tag{6}$$

In terms of visualization, rather than visualizing recommended items in a separate view,⁶⁷ we add recommended nodes to the current node being explored and rearrange the iGraph layout. The end result is that more nodes will show up in the surrounding in a focus+context fashion, and in the meanwhile, we selectively remove some nodes that are less relevant from the layout so that the total number of nodes displayed is kept as a constant. The criteria to remove less relevant nodes could be nodes which have longest paths from the current node being explored, or nodes which are least recently explored, etc.

We also utilize the image popularity and keyword frequency gathered from our data analysis stage for initial suggestions. The display sizes of the popular images are proportional to the logarithmic scale of their popularities. We also set the minimum and maximum display sizes and aspect ratios to avoid very large or small images. The frequent keywords will be highlighted with more saturated colors. The larger display sizes and more saturated colors direct the user's attention for more purposeful exploration.

Algorithm 1 POINT $v = \text{FINDROUTEPOINT}(v_a, v_b, v_m, S_r, \text{distbuff})$ (Refer to Figure 4 (f)).

The line connecting node v_a and v_b overlaps with node v_m

$swap \leftarrow \text{false}$

$v \leftarrow \text{null}$

while $\text{distbuff} > 0$ and v overlaps with nodes in S_r **do**

if $v_a v_b$ intersects with a corner c of v_m **then**

$v \leftarrow c$

else

if $\text{Area}(P) \leq \text{Area}(Q)$ **then**

if $i > j$ **then**

$v \leftarrow (swap ? C_1 : C_3) + \text{distbuff}$

else

$v \leftarrow (swap ? C_2 : C_4) + \text{distbuff}$

else

if $i > j$ **then**

$v \leftarrow (swap ? C_3 : C_1) + \text{distbuff}$

else

$v \leftarrow (swap ? C_4 : C_2) + \text{distbuff}$

if $swap$ **then**

 reduce distbuff

$swap = \neg swap$

return v

6. BUBBLE SET VISUALIZATION

For node comparison and visual recommendation, we assigned different colors to the boundaries of the nodes to indicate their group memberships.⁶⁸ However, the force-directed layout algorithm attracts similar nodes together while pushing dissimilar nodes away. As a result, the nodes belonging to the same group may not be necessarily close to each other. Therefore, users have to go through each individual node to identify its membership. This is very inefficient with a large number of nodes in the graph. It is even more difficult if some nodes have multiple group memberships (Figure 9). To solve these problems, we utilize the bubble set visualization⁶⁹ to highlight the group membership of nodes. Bubble sets use implicit surfaces to create continuous and dynamic hulls. These hulls can be treated as visual containers that highlight group relations without requiring layout adjustments to existing visualizations. Using bubble sets, we can produce tight capture of group members with less ambiguity compared with using convex hulls (e.g., Figures 7 and 10).

Kelp diagram⁷⁰ and KelpFusion⁷¹ can also highlight group memberships. These algorithms first allocate

the space of each node using the Voronoi diagram. Then they both utilize the shortest paths to determine the linkage between nodes in the same group. Kelp diagram simply draws the edges while KelpFusion draws an area instead. We choose to use bubble set because nodes in iGraph have large space occupation while nodes in Kelp diagram and KelpFusion are rather small. In addition, bubble set visualization allows node overlapping, while node overlapping and space allocation are difficult to handle in Kelp diagram and KelpFusion.

In a graph G , assume that the nodes to be highlighted in a bubble set are in the set S_b , and the remaining nodes are in the set S_r . The bubble set should consist of all the nodes in S_b and try to avoid containing or overlapping the nodes in S_r . There are four steps to extract the bubble set. First, we extract a backbone of S_b to connect the nodes. This backbone forms an approximate shape of the final bubble set. Second, for each edge of the backbone that overlaps with the nodes in S_r , we use a routing algorithm to avoid the overlapping. Third, we calculate an energy field to illustrate the display space of the bubble set. A positive value is given to a location near the backbone while a negative value is added if the location is close to the nodes in S_r . Finally, we utilize the marching squares algorithm⁷² to identify the contour of the energy field. The contour should contain all the nodes in S_b and thus becomes the boundary of the bubble set.

6.1 Backbone

The backbone that connects all the nodes in S_b should not only consider the length of the straight line connecting the centers of two nodes but also try to avoid crossing the nodes in S_r . We define the cost of connecting two nodes v_a and v_b in S_b as

$$C(v_a, v_b) = \delta(v_a, v_b) \times (N(v_a, v_b) + 1), \quad (7)$$

where $\delta(v_a, v_b)$ is the length of line segment $v_a v_b$, and $N(v_a, v_b)$ is the number of nodes in S_r that are crossed by $v_a v_b$. We use $(N(v_a, v_b) + 1)$ to make sure that the cost always considers the length of $v_a v_b$. By minimizing the total cost of connecting the nodes in S_b , we generate the backbone. To minimize the cost, Collins et al.⁶⁹ started with the node near the center of S_b and then iteratively added the node with the minimal cost with respect to the selected nodes. The disadvantages of their solution are that it is difficult to define where the center of S_b is and it does not always yield the optimal solution. In contrast, we leverage the minimum spanning tree (MST) to extract the backbone. This eliminates the needs to find the center of S_b and optimizes the solution.

6.2 Routing

Although the backbone extraction tries to choose the edges that overlap with few nodes in S_r , some edges may still cross the nodes in S_r . Routing is then applied to solve this problem. In Figure 4 (a), the backbone edge

$v_a v_b$ intersects with nodes v_m and v_n in S_r . We would add new points to avoid the crossing. In Figure 4 (b), a new point a is added. This point has a distance *distbuff* to the corner of v_m because we would like to reserve some space for the bubble set. However, a is in node v_n , the new edges $v_a a$ and av_b still overlap with node v_n in S_r . Therefore, we place a near the opposite corner of v_m as shown in Figure 4 (c). The new edge $v_a a$ does not overlap with nodes v_m and v_n , but av_b does. This prompts us to add a new point b as shown in Figure 4 (d). The green region illustrates the resulting bubble set. However, if the nodes in S_r are very cluttered, a may still overlap with another node, e.g., v_o as shown in Figure 4 (e). In this case, we will not add new points. The detail routing algorithm is described in Algorithm 1.

6.3 Energy Field and Contour

The energy field illustrates the coverage of the backbone. If a location is close to the backbone, it has a high energy value, otherwise it has a low energy value. If this location is close to the nodes in S_r , the energy value would be reduced or could even be negative. The bubble set is a contour with energy $E(x, y) = c$, where c is a given positive isovalue. This value is selected so that the bubble set will contain all the nodes in S_b and tries to avoid the nodes in S_r . To ensure the performance, we first uniformly partition the display screen into grid points and calculate the energy value for each grid point instead of each pixel. For each grid point P_i , an energy is calculated with respect to an item I_j (i.e., a node or edge) as

$$E(P_i, I_j) = w_j \frac{(D_1 - \delta(P_i, I_j))^2}{(D_1 - D_0)^2}, \quad (8)$$

where D_0 is the distance where the energy is 0, and D_1 is the distance where the energy is the maximum, $\delta(P_i, I_j)$ is the distance between grid point P_i and item I_j , and w_j is the weight assigned to item I_j . The nodes and edges in the backbone have positive weights while the nodes in S_r have negative ones.

Initially, we only calculate the energy of the grid points with respect to the nodes and edges in the backbone. The higher the absolute values of the item weights, the larger the item coverage areas are. Since edges are difficult to observe if they are too thin, in our implementation, we assign a larger value (2) to their weights. The weights of nodes in the backbone are assigned a smaller value 1. If the energy of a grid point is positive, the grid point is close to the backbone. However, the grid points may be close to the nodes in S_r . To avoid the overlapping, we go through the nodes in S_r and add their negative energies to the grid points. The weights of the nodes are assigned the value of -1 . Finally, a positive isovalue c is specified to extract the corresponding contour. A larger value of c may lead to separated components but the contour will get tighter. We will adjust the value of c accordingly so that the contour will form a single connected component.

7. TEXT OUTLIER DETECTION

For an image collection that evolves over time, keywords with high frequencies normally are of interest. However, user interests may shift. Sometimes, an unpopular keyword may appear frequently in a limited time period while a popular keyword may seldom appear or even disappear. Such a phenomenon indicates that an anomalous event happens during that time period. Detecting text outliers can therefore help us gain a deep understanding of the collection by separating them out for further exploration. We exclude K-nodes with low frequencies from this analysis since they already belong to outliers. To detect text outliers with high frequencies, we utilize the algorithm introduced by Akoglu and Faloutsos.⁷³ Their algorithm allows us to detect anomalous time periods and nodes. In order to detect the anomaly, we first evaluate the importance of each node in each time period. The importance of a node can be treated as its current behavior in time period t . Then, a time window w is selected which covers a larger number of time steps than time period t . The importance of each node in w before t can be treated as its recent behavior. By comparing the current and recent behaviors of a node, we can tell how abnormal this node behaves. In addition, by summarizing the abnormality of all the nodes, we know how abnormal this time period is. If the nodes behave similarly to their recent behaviors, then the current time period is normal. Otherwise, it is abnormal.

7.1 Node Importance

Given a time period t , we construct a co-occurrence matrix \mathbf{C}_t to store co-occurrence frequencies of all the K-nodes. According to the Perron-Frobenius theorem,^{74,75} the largest eigenvalue of \mathbf{C}_t is positive and the value for each node in the corresponding eigenvector \mathbf{e}_t indicates the importance of that node.

7.2 Time Period and Keyword Outlier Detection

By measuring the changes of node importance, we can identify anomalous time periods and nodes. We acquire the largest eigenvector of each time period in the time window w before t . Akoglu and Faloutsos⁷³ suggested that the average of these largest eigenvectors (denoted as \mathbf{e}_w) is good enough to evaluate node behaviors. By taking the dot product of the two unit vectors \mathbf{e}_w and \mathbf{e}_t , we get the difference between time period t and its previous time periods in w . This difference value Z is calculated as follows

$$Z = 1 - \mathbf{e}_w^T \mathbf{e}_t. \quad (9)$$

If these two unit vectors are perpendicular to each other, Z is 1 and if they are exactly the same, then Z is 0. The larger the value of Z , the more anomalous time period t is. Given a node, we subtract its corresponding values in \mathbf{e}_w and \mathbf{e}_t to detect how abnormally this node behaves at time period t . After identifying anomalous

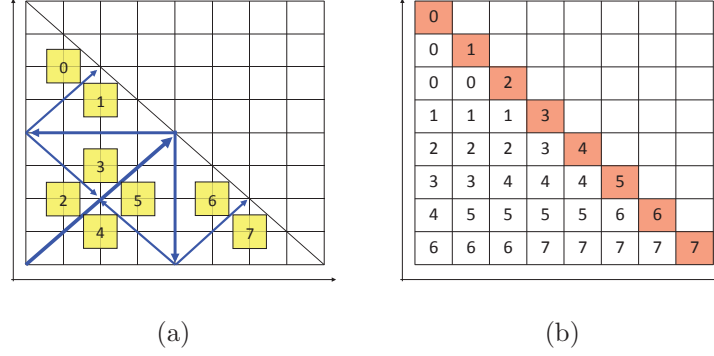


Figure 5. Two different approaches to partition an half matrix, assuming eight GPUs (0 to 7) are used. (a) The first approach partitions the matrix into halves recursively. (b) The second approach works with any number of partitions.

time periods, users can extract related I-nodes and K-nodes for further exploration. If they are interested in an anomalous K-node at a certain time period, they can also use interactive filtering to extract related I-nodes.

8. PARALLEL ACCELERATION AND DISPLAY WALL

To improve the performance of our approach, we leverage a nine-node GPU cluster along with a display wall employed in the Immersive Visualization Studio (IVS) at Michigan Technological University. The GPU cluster consists of one front-end node and eight computing nodes. The display wall consists of 6×4 thin-bezel 46-inch Samsung monitors, each with 1920×1080 pixels. In total, the display wall can display nearly 50 million pixels simultaneously. These 24 monitors are driven by eight computing nodes for computation and visualization. Each node comes with a quad-core CPU, two NVIDIA GeForce GTX 680 graphics cards, and 32GB of main memory. The aggregated disk storage space is over tens of terabytes for the GPU cluster. We use this cluster not only for visualization, but also for performance improvement of preprocessing and layout generation.

8.1 GPU Parallel Preprocessing

For the computation of grayscale and spectrum image distance matrices, it could still take hours to complete when running on a single GPU. To further improve the performance, we use Open MPI and distribute the workload to several GPUs and perform the computation simultaneously.

There are two ways to evenly partition the symmetric distance matrix. As shown in Figure 5 (a), the first approach iteratively partitions the matrix into halves as indicated by blue lines, and the partition results are indicated with yellow rectangles. This approach achieves a perfectly balanced workload while the number of GPUs must be a power of two. Furthermore, since the triangle of each partition is stored as an array in CUDA and the number of images in a collection is an arbitrary number, it is challenging to combine the resulting triangles to the original half matrix in an efficient way.

Figure 5 (b) illustrates the second approach. Assuming that we need to partition the half matrix into p partitions, we first divide the matrix into p rows and p columns. Then for the p rectangles along the diagonal, each partition gets one. The number of remaining rectangles is $p \times (p-1)/2$, which means that each partition gets $(p-1)/2$ rectangles. If p is an odd number, then each partition gets the same number of rectangles. Otherwise, half of the partitions get one more rectangle than the rest. Although the number of images for each partition to load is not perfectly balanced, this approach still achieves a very balanced workload for a large p and it works well with any number of GPUs. In practice, p could be a large number (e.g., a multiple of the number of GPUs) and we can distribute the partitions to each GPU in a round robin manner. Meanwhile, because we store the computation results for each partitioned rectangle, it is easy to index and retrieve the similarity value for any pair of images. Therefore, we implement the second approach.

8.2 CPU Parallel Graph Layout

We use MPI for the parallel iGraph layout generation using multiple CPUs. The most time-consuming computations are initial backbone iGraph, constrained layout adjustment, and density adjustment. Thus we parallelize these three steps to achieve good performances. The input is the information of all I-nodes and K-nodes (i.e., their initial locations, widths, and heights) while the output is the iGraph layout after applying these steps.

For initial backbone iGraph, each node needs to calculate the repulsive force with any other node. After we distribute the graph nodes evenly to the processors, they need to exchange position information in each iteration of the FR algorithm. The initial layout changes dramatically in early iterations and gets more and more stable in later iterations. To reduce the communication cost, we allow the processors to have more frequent communications in early iterations than later ones. For constrained layout adjustment, there are four forces. The bidirectional and unidirectional repulsive forces may occur between any two nodes. Therefore, after we distribute the graph nodes to different processors, they also need to exchange position information in each iteration. Our experience shows that the attractive and spring forces take much more time to compute than the two repulsive forces. As such, when the processors have similar numbers of attractive and spring forces, we consider that the workload balancing is achieved. Since the numbers of attractive and repulsive forces are related to the degrees of nodes, we sort node degrees in the decreasing order and allocate those graph nodes to different processors in a round robin manner. A processor would not receive any more nodes if the accumulated node degrees reaches the average total node degrees it should have. In this way, all the processors shall have a similar amount of accumulated node degrees. Similar to constrained layout adjustment, in density adjustment, we distribute K-nodes based on their degrees. In each iteration, after calculating the new positions of the K-nodes, the processors exchange position information, update the densities of the triangles, and repeat the process until certain criteria are met.

data set	# images	# keywords	# I-I edges	# K-K edges	# I-K edges
APOD	4,560	5,831	10M	17M	137K
MIR Flickr	25,000	19,558	312M	191M	173K

Table 1. The sizes of APOD and MIR Flickr data sets (M: million, K: thousand).

data set	# images	similarity type	block config.	thread config.	loading GPU	computing GPU	speedup factor	saving GPU	# graph nodes	layout CPU	speedup factor
APOD	4,560	grayscale	16,384	512	0.214s	762.312s	-	0.008s	125	0.38s	-
		spectrum	16,384	1,024	0.213s	515.042s	-	0.007s	225	1.29s	-
		histogram	16,384	1,024	0.002s	3.259s	-	0.008s	525	6.96s	-
MIR Flickr	25,000	grayscale	512	512	12.715s	25,722s	-	0.312s	550	10.24s	-
			768	768	3.179s*	3,127s*	8.22	0.088s*		0.90s*	11.37
		spectrum	512	512	12.623s	13,380s	-	0.272s	1050	47.60s	-
			768	768	3.163s*	2,151s*	6.22	0.089s*		2.07s*	22.99
		histogram	1,024	1,024	0.0296s	0.117s	-	0.274s	2050	5.61s*	-

Table 2. Parameter values and timing results. The GPU’s loading and saving are copying the data from and back to the CPU. The CPU layout time includes the time for workload assignment, data copying, layout computation (150 iterations, this step dominates the timing), and communication of node information. A * indicates the timing with multiple processors (8 GPUs for distance computation, 24 CPUs for layout generation). Otherwise, the timing is on a single processor.

8.3 Display Wall Visualization

To render iGraph on the display wall, we design four program components: the *master*, *computation*, *relay* and *slave* programs. The master program not only has a user interface to accept all the user interactions but also displays iGraph. It runs on a local computer which is located in the same room as the display wall. This computer captures user interactions, sends instructions to the computation program for generating the graph layout in parallel. After receiving the layout result from the computation program, the master program sends data to the cluster’s front-end node. The data contain all the information for rendering, e.g., all the I-nodes and K-node positions, widths and heights. Meanwhile, this front-end node running the relay program receives the data and broadcasts it to the eight computing nodes of the cluster. The slave program is an OpenGL application running on these eight nodes. Each node receives the data sent from the relay program, decodes the data, and renders the visualization results to the three tiles which it is responsible for.

For the slave program, there are two problems that we need to address. The first is that the time gap between neighboring receiving actions is less than that of decoding so that the data may change undesirably during decoding. The second is the synchronization of receiving, decoding and displaying because the data may change if two of these three actions happen simultaneously. To address the first problem, we create a large buffer, use one thread to receive the data and save it to the buffer so that the new incoming data will not override the previous data. This also prevents receiving from interrupting decoding and rendering. Furthermore, to solve the synchronization of decoding and rendering, we create another thread to iteratively decode the data and invoke

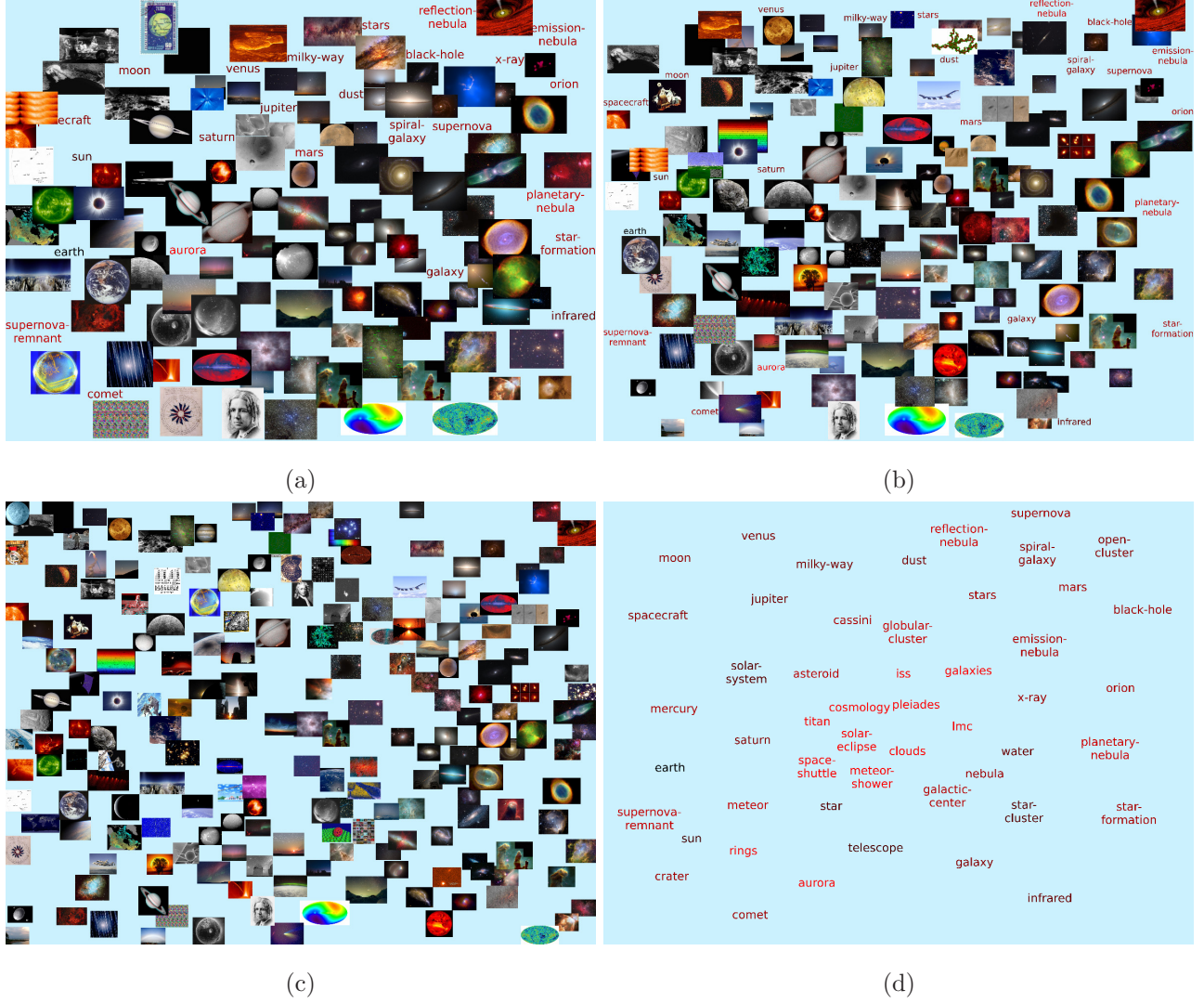


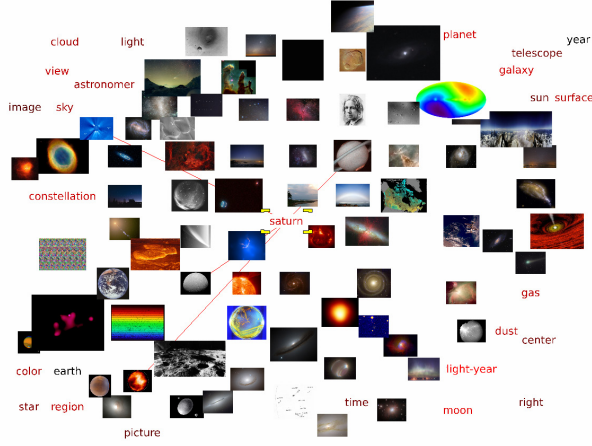
Figure 6. iGraphs with different numbers of I-nodes and K-nodes. For (a) and (b), the numbers of I-nodes and K-nodes are 110 and 25, 190 and 25, respectively. (c) and (d) show the single graphs with 270 I-nodes and 50 K-nodes, respectively.

OpenGL to display. In our experiment, we did not encounter inconsistency among the eight computing nodes. Therefore, we do not synchronize the tasks among them.

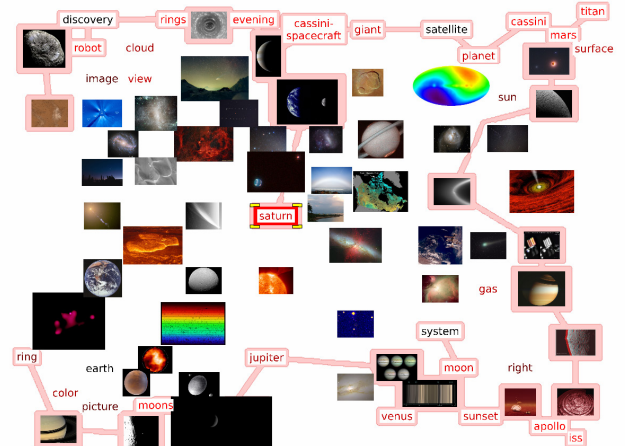
9. RESULTS

9.1 Data Sets and Performance

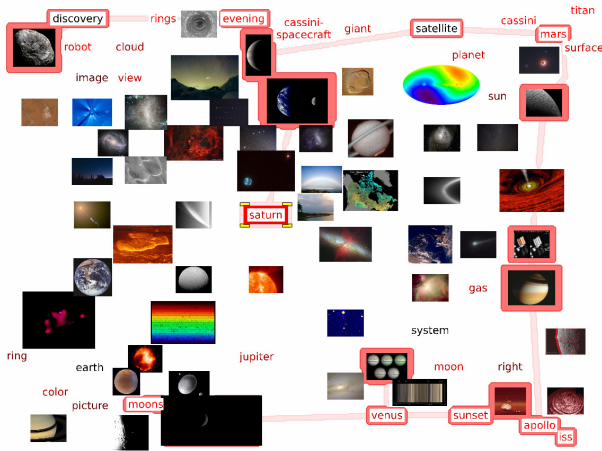
In our experiments, we used two data sets, APOD and MIR Flickr. Their data sizes are shown in Table 1. We can see that the connections between I-nodes and K-nodes are less in MIR Flickr than in APOD. On average there are only 6.9 K-nodes connected to each I-node in MIR Flickr while 31.1 in APOD. This results in two distinguished groups of I-nodes and K-nodes in the initial iGraph for the MIR Flickr data set. The configurations



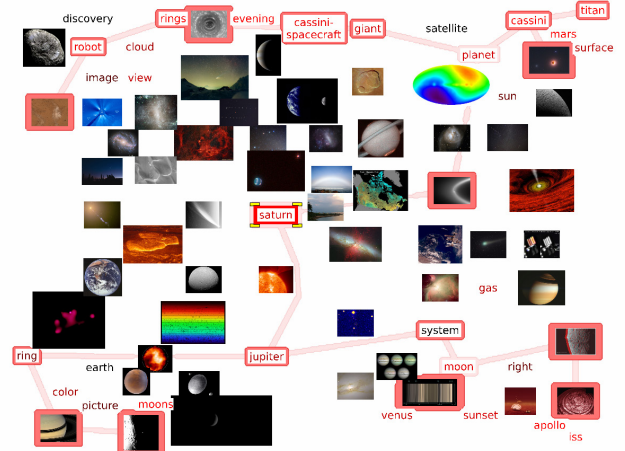
(a)



(b)



(c)



(d)

Figure 7. Visual recommendation of the APOD data set. (a) and (b) iGraphs before and after the recommendation (based on K-node “saturn”), respectively. All the recommended nodes are highlighted with the bubble set in (b). The nodes recommended by the user-centric and primitive item-centric approaches are highlighted in (c) and (d), respectively. Those nodes highlighted with dark red boundaries in (c) and (d) do not show up in the previous iGraph.

and timing results for computing image distance matrices are shown in Table 2. The single GPU computation time was reported using a desktop PC (not a cluster node) with an nVIDIA GeForce GTX 580 graphics card, while multiple GPUs computation time was reported using the GPU cluster (Section 8). At run time, all the tasks and interactions are interactive. In the following, we present iGraph results with screenshots captured from our program. For iGraph interaction and its running on the display wall, please refer to the supplementary video.

	APOD					MIR Flicker				
# of recommended nodes	12	25	50	100	200	12	25	50	100	200
user-centric	7	7	9	5	0	1	5	4	0	0
primitive item-centric	6,457	6,012	4,670	1,528	49	33,572	29,152	22,728	17,524	8,015
rooted PageRank	6,942	6,515	6,192	4,769	1,227	36,580	38,130	39,797	37,083	29,422
Katz algorithm	638	1,266	2,855	6,989	9,672	1,105	3,070	6,050	8,325	14,843

Table 3. Comparing four recommendation approaches. Each approach recommends 12 to 200 nodes for all the I-nodes and K-nodes in the data sets. The frequencies of each approach that preforms the best are listed. For each case, the highest frequency is highlighted in bold.

(c) and (d) are the single graphs with only I-nodes and K-nodes, respectively. Single graphs are displayed when users want to observe only I-I edge or K-K edge relationships.

9.3 Visual Recommendation

The initial backbone iGraph only gives an overview while our recommendation system serves as a navigation tool to help users in their exploration. The example with the APOD data set is shown in Figure 7. From (c) we know that users who are interested in “saturn” are also interested in “mars”, “moons”, and “satellite”. In (d), for the keywords in the bubble set, “cassini” and “cassini spacecraft” correspond to the Cassini orbiter which reached Saturn and its moons in 2004. “ring” means the outer ring of Saturn.

Another example with the MIR Flickr data set is shown in Figure 8. In (b), we show iGraph after recommendation based on an image of a fox. From (c) we know that the users who are interested in the fox are also interested in “furry” animals such as squirrel and otter. In (d), “red”, “green”, “garden”, and “fox” are related to the image of the fox.

The user-centric approach and the primitive approach have their own limitations. When we either do not have many users or do not have much use exploration history, the user-centric approach sometimes could not recommend any node since the selected node was not explored by any user previously. If an I-node (K-node) does not have many related K-nodes (I-nodes), the primitive approach also could not recommend enough nodes. To overcome the limitations, we implement the rooted PageRank and Katz algorithms. These two algorithms connect all the I-nodes and K-nodes together as a graph. The similarity between two nodes depends on their closeness in the graph. Therefore, two nodes that share a large number of related nodes will be similar. This is not the case for the primitive approach. In addition, since all the nodes are connected together, we can always recommend enough nodes for any selected node.

To compare the performances of these four approaches, we go through each node in iGraph and make recommendation based on that node. For any given node v_a used for recommendation, if a node v_b recommended by an approach is also recommended by any other approach, we consider v_b as a valid recommendation. The



Figure 9. Node comparison of (a) the APOD data set and (b) the MIR Flickr data set. (a) Three K-nodes “emission nebula”, “planetary nebula”, and “reflection nebula” are chosen for comparison. The nodes with green bottom-left corners, red upper-left corners, and blue upper-right corners are related to “emission nebula”, “planetary nebula” and “reflection nebula”, respectively. (b) One K-node and two I-nodes are chosen for comparison. The K-node is “self-portrait” and the two I-nodes are images of people. One is a picture of a girl wearing a T-shirt and the other is a picture of a pair of feet.

approach that has the largest number of valid recommendations is the best for v_a . Table 3 lists the frequencies of the best approaches for the APOD and MIR Flickr data sets. From the table, we can observe the following four facts. First, the user-centric approach performs the worst simply because we do not have large user histories. Second, among the ten test cases, the rooted PageRank performs the best except for two cases with the APOD data set, for which the Katz algorithm performs the best. Third, the Katz algorithm performs better as the number of recommended nodes increases. This is due to the fact that this approach has very strict requirements. To measure the similarity of two nodes, it considers not only their closeness, but also the number of nodes which are close to both nodes. Since the requirement is so strong, the most similar nodes based on the Katz algorithm are not recommended by the other approaches. On the other hand, since the Katz algorithm considers so much information, the similarity measurement gets more accurate as the number of recommended nodes increases. Fourth, the primitive approach and rooted PageRank algorithm perform similarly. However, the performance of the primitive approach drops dramatically with the increasing number of nodes recommended. That is because as the number of recommended nodes increases, the primitive approach could not recommend enough nodes. Therefore, the number of valid recommendations decreases. Since none of the approaches work the best for all the cases, we suggest to use the most valid recommendations (i.e., nodes that recommended by the most number of approaches) as the recommendation results.

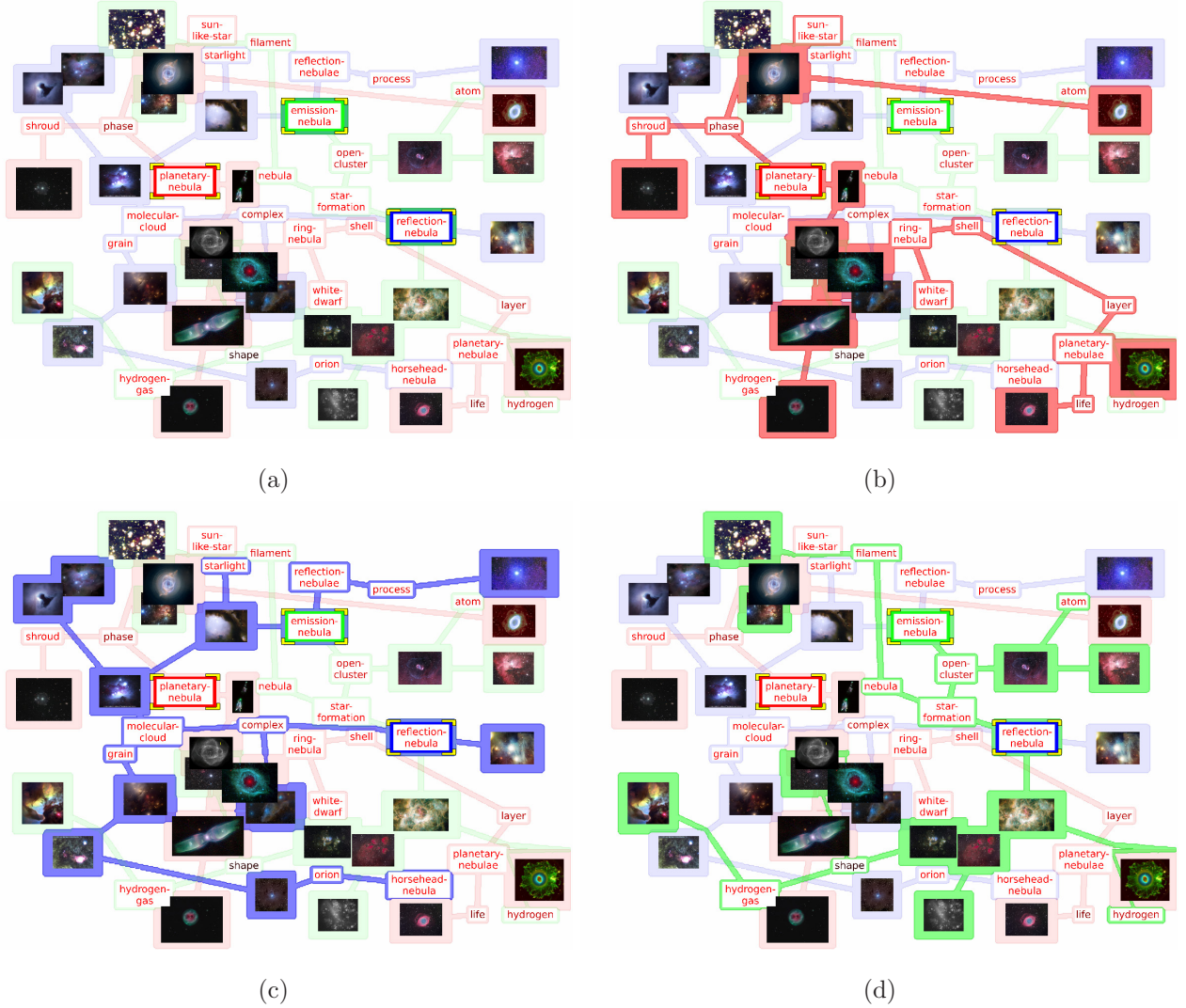


Figure 10. Node comparison of Figure 9 (a) using the bubble set visualization to highlight the belonging relations. (a) shows all three groups of nodes with light boundary colors while (b) to (d) highlight each group of nodes with darker boundary colors.

9.4 Node Comparison

While visual recommendation helps users explore similar nodes or nodes clicked by others, node comparison allows users to choose multiple nodes for detailed comparison. For example, a user interested in the three types of nebulas would choose keywords “emission nebula”, “reflection nebula”, and “planetary nebula”. The comparison result is shown in Figure 9 (a). The images provide the user with a brief impression of what these nebulas look like. The images of nebulas with green bottom-left corners are “emission nebula”. Most of them are reddish because they are clouds in high temperature and emit radiation. The images of nebulas with blue upper-right corners are “reflection nebula”. They are bluish and darker because they are efficient to scatter the blue light.



Figure 11. Node comparison of Figure 9 (b) using the bubble set visualization to highlight the belonging relations. (a) shows all three groups of nodes with light boundary colors while (b) to (d) highlight each group of nodes with darker boundary colors.

The images of nebula that look like planes and have red upper-left corners are “planetary nebula”. Meanwhile, the user could also get knowledge from the other keywords. For example, “white dwarf” and “layer” are linked with “planetary nebula” because when an aged red supergiant collapses, the outer layer is the “planetary nebula” while the inside core is the “white dwarf”. However, the nodes in the same group may not always be close to each other. As a result, users need to go through all the nodes to figure out their belonging relations. Figure 10 (a) highlights the belonging relations with the bubble sets. When users prefer to study the nodes related to a selected node, they will be further highlighted with dark boundaries as shown in Figure 10 (b), (c), and (d).

In another example, we choose one keyword “self-portrait” and two images of people. The comparison result

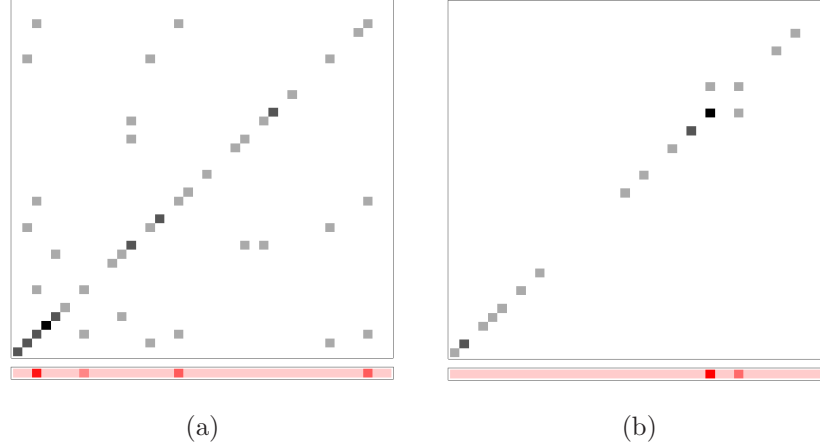


Figure 12. Co-occurrence matrices and their corresponding eigenvectors of the APOD data set in (a) March 2001 and (b) April 2001. The 40 most frequent keywords from the entire data set are used to construct each matrix. Gray rectangles represent non-zero values with darker colors indicating higher frequencies. The corresponding eigenvector is below the co-occurrence matrix where darker red indicates higher values.

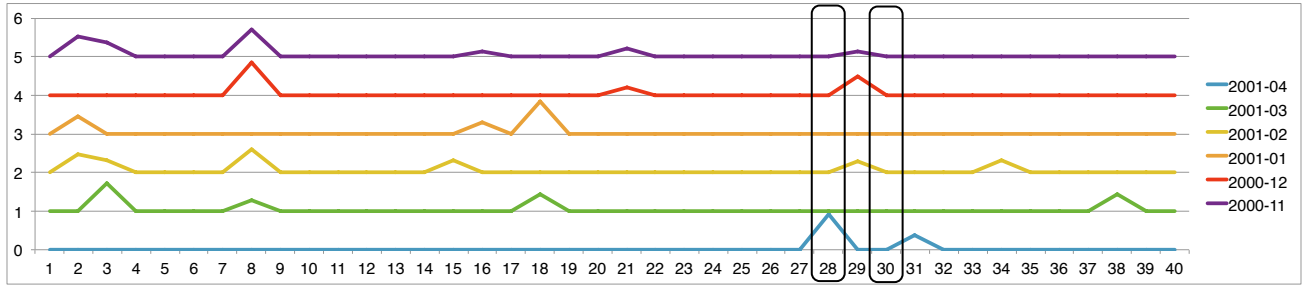


Figure 13. The eigenvectors of the 40 most frequent keywords from November 2000 to April 2001. The significant difference is the dramatic jump of importance value for keywords “space shuttle” (index 28) and “iss” (index 31). The interest shift to the international space station and space shuttle is due to the launching of the space shuttle on April 23, 2001.

is shown in Figure 9 (b). The keyword “365 days” is marked by three different colors, which means that it is related to the K-node and two I-nodes chosen. “me”, “ofme”, and “self” are highlighted with both red and blue boundaries which means that they are related to the keyword “self-portrait” and the image showing a pair of feet. Keywords “red shoes” and “square” with blue boundaries indicate that there is a pair of shoes in the picture and the photo was taken at a square. Figure 11 (a) shows the same result using the bubble set visualization. The K-nodes “365 days”, “me”, “ofme”, and “self” are highlighted with blended colors to indicate that they are related to multiple selected nodes. Figure 11 (b), (c) and (d) are the results of highlighting the related nodes of a selected node.



Figure 14. Photos showing iGraph of the MIR Flickr data set running on the 24-tile display wall (1000 images and 50 keywords are displayed).

9.5 Text Outlier Detection

Figures 12 and 13 demonstrate an anomalous time period and its anomalous K-nodes. Figures 12 (a) and (b) show the co-occurrence matrices and their corresponding eigenvectors of the APOD data set in March and April of 2001, respectively. The 40 most frequent keywords in the meta-tagged keywords are used to construct each matrix. These keywords are selected because they appear at least 50 times in the entire data set. The gray rectangles in Figure 12 represent the non-zero values in the co-occurrence matrix. Darker colors indicate higher frequencies. The corresponding eigenvector is shown below the co-occurrence matrix, where darker red indicates higher values. Many gray rectangles spread in Figure 12 (a), which indicates no distinct interest in March 2001. The four darker red rectangles in the eigenvector indicate four important K-nodes: “saturn”, “jupiter”, “spacecraft”, and “rings”. The co-occurrence matrix and its corresponding eigenvector in Figure 12 (b) allow us identify the most important keywords in April 2001: “space shuttle” and “iss” (international space station). Figure 13 lists the eigenvectors of the APOD data set within the five months before April 2001. The outlier detection function indicates that the APOD collection behaved differently in April 2001 compared with its previous five months. The abnormal K-nodes indicate the interest shifted to “space shuttle” and “international space station” since the two keywords were not important in the previous months but became very important in April 2001. This interest shifting may be caused by the launching of the space shuttle on April 23, 2001. Closer examination shows that the APOD creators not only posted figures related to the space shuttle, but also posted some related topics, such as the first man entering the space, the first space quidditch match between the United States and Russia, and STS-1 (the first shuttle launch).

9.6 Running on Display Wall

Figure 14 shows three iGraph photos of the MIR Flickr data set rendered on the display wall. With the display wall, we are able to display thousands of images and keywords simultaneously for comfortable viewing. Currently, we are using this display wall for showing iGraph demos to visitors, including university alumni, visitors, and

summer students. Initial feedback from several groups of visitors is fairly positive as they commented that running iGraph on this life-size tiled display is much more expressive and fun to watch compared with running on a regular desktop display. The advantage of using the display wall is that it allows more than a dozen of people to comfortably view and discuss the results together in such a collaborative environment. Nevertheless, with the dramatic expanding of display area, it takes more effort for a viewer to correlate and compare images that are on the opposite sides of the display wall, especially for those images close to the wall’s boundary.

10. EVALUATION

We recruited five unpaid researchers in astronomy for a user study of iGraph on a PC: one professor, three PhD students, and one master student. Four of them are researchers in physics studying astronomy and the remaining one is a computer science PhD student with a master in physics focusing on astronomy. The user study was conducted in a lab using the same PC. The PC has a 27-inch monitor with 1920×1080 resolution, where the visualization result occupied an area of 1600×900 . Users were first introduced to iGraph about its goal and main functions. Then they were given the APOD data set for free exploration to get familiar with the system.

The iGraph survey includes seven tasks and five general questions. The seven tasks (T1 to T7) were designed to evaluate the effectiveness of iGraph. T1 and T2 asked the user to compare the compound graph with the single graphs and select the numbers of images and keywords that she was satisfied with for the initial backbone graph. T3 asked the user to go over all the images and keywords to get an impression of the APOD data set. T4 and T5 asked the user to evaluate the recommendation function (user- and primitive item-centric approaches) with a given example and based on her preference, respectively. T6 and T7 asked the user to evaluate the comparison function with a given example and based on her preference, respectively. Every task has several survey questions and write-in questions except T2 which consists of only write-in questions. All the survey questions have the same set of choices on a 5-point Likert scale: 1: strongly disagree, 2: disagree, 3: neutral, 4: agree, and 5: strongly agree.

We summarized the questions of the seven tasks as shown in Table 4. For simplicity, if the user rated the question as 4 or 5, we consider that she agreed; if she rated it as 1 or 2, she disagreed; otherwise, she was neutral. We used the number of users that agreed to measure the effectiveness of iGraph functions.

Users could perform the tasks whenever they felt comfortable. Each study took about one hour to complete. Although each task could be performed within a few minutes, users frequently returned to the interface for further verification when writing their comments, which took most of the time. Users were informed that the tasks were not timed and their comments were of crucial importance. Among these tasks, two of them are related to the free explorations of the data set with recommendation and comparison functions. Each of these

task	description	summarized question(s)	average rating
T1	Compare the compound graph with the single graphs	Was iGraph helpful in showing the relationship among images and keywords?	5.0
T2	Select the numbers of images and keywords that the user is satisfied for the initial backbone graph	Were the images more useful than the keywords?	4.0
T3	Go over all the images and keywords to get an impression of the APOD data set	Were the functions that allow the user to go through all images (keywords) helpful?	4.0 (5.0)
T4	Evaluate the visual recommendation function with a given example	Was the visual recommendation function helpful with respect to the given example?	5.0
T5	Evaluate the visual recommendation function based on user preferences	Was the visual recommendation function helpful with respect to user preferences?	5.0
T6	Evaluate the node comparison function with a given example	Was the node comparison function helpful with respect to the given example?	5.0
T7	Evaluate the node comparison function based on user preferences	Was the node comparison function helpful with respect to user preferences?	5.0

Table 4. Seven tasks in the user study and users’ response to summarized questions.

two tasks took them a little more time till they were ready to write comments. The outcome of the user study is summarized in Table 4.

Next we provide further explanation on these outcomes. For T1 and T2, we asked users to shift freely between the single graphs and compound graph. In addition, they were asked to change the numbers of images and keywords displayed on the screen in order to get an impression of the best ratio between images and keywords. In terms of T1, the feedback was positive. All users stated that iGraph was helpful to show the relationships among images and keywords in both compound and single graphs. Furthermore, a compound graph was considered better than a single graph. The animation of changing between compound and single graphs was helpful to gain an impression of node changes. In terms of T2, one PhD student preferred more keywords than images because the keywords were easier to understand and intuitive to query. The rest of users indicated that since images were more attractive, the ratio between the numbers of images and keywords should be within the range of 3 : 1 to 10 : 1.

For T3, we asked users to go through all the images and keywords to gain an impression of the APOD collection. All the images were shown in the order of their published dates and all the keywords were sorted based on their frequencies. Four users considered the functions that allow the user to go through all images were helpful to gain an overall impression of the data set and to select images of interest. The remaining user was neutral on this since she was not familiar with all images and viewing individual images did not make much sense to her. For better browsing of the collection, we can group images based on their similarities and show them one group at a time. Moreover, all users indicated that going through all keywords was helpful. Three agreed that iGraph could be helpful to identify popular and rare keywords while two were neutral on it. This was probably

due to the equal weights of the keywords extracted from the explanation and keyword sections. Since the number of keywords extracted from the explanation section is larger than that from the keyword section, equal weights lead to the reduced importance of the keywords from the keyword section. Finally, all users agreed that color and opacity encoding was helpful to identify I-nodes and K-nodes already in iGraph so that they could select new nodes into iGraph for recommendation and comparison.

For T4, users were asked to type keyword “earth” for recommendation and see the images and keywords recommended using different approaches. For T5, users could select any I-node or K-node from iGraph for recommendation and repeat this process as long as they wanted to. All users agreed that the recommendation function was useful, especially for the primitive item-centric approach because it recommended more relevant nodes than the user-centric approach. However, we believe that the user-centric approach will recommend more relevant nodes when the number of users increases to a certain level. The functions that allow the user to add any image and keyword made the recommendation function more effective. They also believed that the highlighted boundaries were helpful to identify the recommended nodes.

For T6, users were asked to select an image related to “jupiter” and the keyword “moon” for comparison. For T7, users could select up to four nodes to compare, and then select any image or keyword in the collection for further exploration. All users agreed that the comparison function helped them find the nodes related to one or multiple selected nodes. Three of them could further dig out the underlying relationships of the selected nodes while two could not. This was because we linked both images and keywords to the selected nodes to show their relationships, which was not a direct approach. All of them agreed that the highlighted boundaries could help identify the belonging relationships of the nodes. Two of them thought that the animation could not demonstrate node changes. This was because when users selected multiple nodes for comparison, most of the nodes after comparison did not appear before. Thus, the animation only gave them the impression of layout change.

For the general questions, two users mentioned that iGraph could not only support researchers to find the underlying relationships of the APOD data set but also help teachers teach in class. Besides that, they stated that iGraph could be extended to handle other image or image and text collections (such as movie collections) and to assist in image search. Users also pointed out some possible improvements for iGraph. First, the techniques related to object identification could be helpful for measuring image similarity. Second, they suggested us to provide more sorting options while going through the APOD collections, e.g., sorting according to the image colors, alphabetical order of keywords, or user preference. Third, they also mentioned that we could provide more flexible query functions. Therefore, we provided text outlier detection to identify the outliers in a certain time period to help users narrow down to certain keywords of interest during the query. Finally, one suggested

that we should show the detailed information of a selected image.

11. CONCLUSIONS AND FUTURE WORK

We have presented iGraph, a new approach to visualize large image collections associated with text information. iGraph provides essential capabilities including interactive filtering, node comparison, and visual recommendation. The general approach proposed for iGraph is applicable to many other large graph drawing, visualization, and exploration applications. We integrate additional features including the use of bubble sets, suggestion of abnormal keywords and time periods, and comparison of different recommendation solutions to enrich the power of iGraph. We leverage multiple GPUs for preprocessing and multiple CPUs for runtime graph layout generation and visualization. Our user evaluation shows that iGraph is a promising approach for navigating and viewing large image and text collections.

The current implementation of iGraph also has some limitations. First, we do not automatically update the similarity matrices when the collection gets larger or keeps evolving. An efficient solution that supports incremental update should be sought. Second, the similarities between images are computed based on low-level pixel information. High-level object identification algorithms may provide more accurate measurement. Third, color blending in bubble set visualization does not best show the belonging relations of the nodes that belong to multiple clusters. Further research is needed to fully address this problem. We plan to solve these limitations and develop an online version of iGraph to benefit a wide user community.

ACKNOWLEDGMENTS

This research was supported in part by the U.S. National Science Foundation through grants IIS-1017935, IIS-1456763, IIS-1455886, and the Notre Dame Andrónico Luksic Grants Program. Denis Parra was supported by FONDECYT Grant 11150783. We thank the anonymous reviewers for their helpful comments. We also thank Mark Huiskes, Bart Thomee, and Michael Lew for making the MIR Flickr collection available. All the images at the APOD website are credited to the owners or institutions where they originated.

REFERENCES

- [1] Nemiroff RJ, Bonnell JT. Astronomy Picture of the Day: <http://antwrp.gsfc.nasa.gov/apod/astropix.html>. In: Bulletin of the American Astronomical Society; 1995. p. 1291.
- [2] Huiskes MJ, Lew MS. The MIR Flickr Retrieval Evaluation. In: Proceedings of ACM International Conference on Multimedia Information Retrieval; 2008. p. 39–43.
- [3] Chen C, Gagaudakis G, Rosin P. Content-Based Image Visualization. In: Proceedings of International Conference on Information Visualisation; 2000. p. 13–18.

- [4] Bederson BB. PhotoMesa: A Zoomable Image Browser using Quantum Treemaps and Bubblemaps. In: Proceedings of ACM Symposium on User Interface Software and Technology; 2001. p. 71–90.
- [5] Platt JC, Czerwinski M, Field BA. PhotoTOC: Automatic Clustering for Browsing Personal Photographs. In: Proceedings of IEEE Pacific Rim Conference on Multimedia; 2003. p. 6–10.
- [6] Torres R, Silva C, Medeiros C, Rocha H. Visual Structures for Image Browsing. In: Proceedings of International Conference on Information and Knowledge Management; 2003. p. 49–55.
- [7] Jankun-Kelly TJ, Ma KL. MoireGraphs: Radial Focus+Context Visualization and Interaction for Graphs with Visual Nodes. In: Proceedings of IEEE Information Visualization Symposium; 2003. p. 59–66.
- [8] Yang J, Fan J, Hubball D, Gao Y, Luo H, Ribarsky W, et al. Semantic Image Browser: Bridging Information Visualization with Automated Intelligent Image Analysis. In: Proceedings of IEEE Symposium on Visual Analytics Science and Technology; 2006. p. 191–198.
- [9] Brivio P, Tarini M, Cignoni P. Browsing Large Image Datasets through Voronoi Diagrams. *IEEE Transactions on Visualization and Computer Graphics*. 2010;16(6):1261–1270.
- [10] Wang C, Reese JP, Zhang H, Tao J, Nemiroff RJ. iMap: A Stable Layout for Navigating Large Image Collections with Embedded Search. In: Proceedings of IS&T/SPIE Conference on Visualization and Data Analysis; 2013. .
- [11] Wang C, Reese JP, Zhang H, Tao J, Gu Y, Ma J, et al. Similarity-Based Visualization of Large Image Collections. *Information Visualization*. 2015;14(3):183–203.
- [12] Zhang H, Tao J, Ruan F, Wang C. A Study of Animated Transition in Similarity-Based Tiled Image Layout. *Tsinghua Science and Technology*. 2013;18(2):157–170.
- [13] Šilić A, Basić BD. Visualization of Text Streams: A Survey. In: Proceedings of International Conference on Knowledge-Based and Intelligent Information and Engineering Systems: Part II; 2010. p. 31–43.
- [14] Alencar AB, de Oliveira MCF, Paulovich FV. Seeing Beyond Reading: A Survey on Visual Text Analytics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*. 2012;2(6):476–492.
- [15] Gan Q, Zhu M, Li M, Liang T, Cao Y, Zhou B. Document Visualization: An Overview of Current Research. *Wiley Interdisciplinary Reviews: Computational Statistics*. 2014;6(1):19–36.
- [16] Wanner F, Stoffel A, Jäckle D, Kwon BC, Weiler A, Keim DA. State-of-the-Art Report of Visual Analysis for Event Detection in Text Data Streams. In: Eurographics Conference on Visualization - State of The Art Reports; 2014. .
- [17] Kucher K, Kerren A. Text Visualization Techniques: Taxonomy, Visual Survey, and Community Insights. In: Proceedings of IEEE Pacific Visualization Symposium (Visualization Notes); 2015. p. 117–121.
- [18] Wattenberg M, Viegas FB. The Word Tree, An Interactive Visual Concordance. *IEEE Transactions on Visualization and Computer Graphics*. 2008;14(6):1221–1228.

- [19] Clarkson E, Desai K, Foley JD. ResultMaps: Visualization for Search Interfaces. *IEEE Transactions on Visualization and Computer Graphics*. 2009;15(6):1057–1064.
- [20] Viégas FB, Wattenberg M, Feinberg J. Participatory Visualization with Wordle. *IEEE Transactions on Visualization and Computer Graphics*. 2009;15(6):1137–1144.
- [21] Koh K, Lee B, Kim B, Seo J. ManiWordle: Providing Flexible Control over Wordle. *IEEE Transactions on Visualization and Computer Graphics*. 2010;16(6):1190–1197.
- [22] Cui W, Wu Y, Liu S, Wei F, Zhou MX, Qu H. Context Preserving Dynamic Word Cloud Visualization. In: *Proceedings of IEEE Pacific Visualization Symposium*; 2010. p. 121–128.
- [23] Gambette P, Véronis J. Visualising a Text with a Tree Cloud. In: *Proceedings of International Federation of Classification Societies Conference*; 2010. p. 561–569.
- [24] Lee B, Riche NH, Karlson AK, Carpendale S. SparkClouds: Visualizing Trends in Tag Clouds. *IEEE Transactions on Visualization and Computer Graphics*. 2010;16(6):1182–1189.
- [25] Yang J, Luo D, Liu Y. Newdle: Interactive Visual Exploration of Large Online News Collections. *IEEE Computer Graphics and Applications*. 2010;30(5):32–41.
- [26] Havre S, Hetzler E, Whitney P, Nowell L. ThemeRiver: Visualizing Thematic Changes in Large Document Collections. *IEEE Transactions on Visualization and Computer Graphics*. 2002;8(1):9–20.
- [27] Wei F, Liu S, Song Y, Pan S, Zhou MX, Qian W, et al. TIARA: A Visual Exploratory Text Analytic System. In: *Proceedings of ACM SIGKDD Conference*; 2010. p. 153–162.
- [28] Breuel TM, Janssen WC, Popat K, Baird HS. Paper to PDA. In: *Proceedings of IRPA International Conference on Pattern Recognition*; 2002. p. 476–479.
- [29] Berkner K, Schwartz EL, Marle C. SmartNails: Display- and Image-Dependent Thumbnails. In: *Proceedings of SPIE Conference on Document Recognition and Retrieval*; 2003. p. 54–65.
- [30] Strobel H, Oelke D, Rohrdantz C, Stoffel A, Keim DA, Deussen O. Document Cards: A Top Trumps Visualization for Documents. *IEEE Transactions on Visualization and Computer Graphics*. 2009;15(6):1145–1152.
- [31] Fruchterman TMJ, Reingold EM. Graph Drawing by Force-Directed Placement. *Software - Practice and Experience*. 1991;21(11):1129–1164.
- [32] Kamada T, Kawai S. An Algorithm for Drawing General Undirected Graphs. *Information Processing Letters*. 1989;31(1):7–15.
- [33] Davidson R, Harel D. Drawing Graphs Nicely Using Simulated Annealing. *ACM Transactions on Graphics*. 1996;15(4):301–331.
- [34] Herman I, Melancon G, Marshall MS. Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*. 2000;6(1):24–43.

- [35] Díaz J, Petit J, Serna M. A Survey of Graph Layout Problems. *ACM Computing Surveys*. 2002;34(3):313–356.
- [36] Hachul S, Jünger M. An Experimental Comparison of Fast Algorithms for Drawing General Large Graphs. In: *Proceedings of International Symposium on Graph Drawing*; 2006. p. 235–250.
- [37] Pienta R, Abello J, Kahng M, Chau DH. Scalable Graph Exploration and Visualization: Sensemaking Challenges and Opportunities. In: *Proceeding of International Conference on Big Data and Smart Computing*; 2015. p. 271–278.
- [38] Walshaw C. A Multilevel Algorithm for Force-Directed Graph Drawing. In: *Proceedings of International Symposium on Graph Drawing*; 2000. p. 171–182.
- [39] Holten D, van Wijk JJ. Force-Directed Edge Bundling for Graph Visualization. *Computer Graphics Forum*. 2009;28(3):983–990.
- [40] Zinsmaier M, Brandes U, Deussen O, Strobel H. Interactive Level-of-Detail Rendering of Large Graphs. *IEEE Transactions on Visualization and Computer Graphics*. 2012;18(12):2486–2495.
- [41] Frishman Y, Tal A. Multi-Level Graph Layout on the GPU. *IEEE Transactions on Visualization and Computer Graphics*. 2007;13(6):1310–1317.
- [42] Bastian M, Heymann S, Jacomy M. Gephi: An Open Source Software for Exploring and Manipulating Networks. In: *Proceedings of AAAI International Conference on Weblogs and Social Media*; 2009. .
- [43] Alper B, Höllerer T, Kuchera-Morin J, Forbes A. Stereoscopic Highlighting: 2D Graph Visualization on Stereo Displays. *IEEE Transactions on Visualization and Computer Graphics*. 2011;17(12):2325–2333.
- [44] Dwyer T, Marriott K, Stuckey PJ. Fast Node Overlap Removal. In: *Proceedings of International Symposium on Graph Drawing*; 2005. p. 153–164.
- [45] Gansner ER, Hu Y. Efficient Node Overlap Removal Using a Proximity Stress Model. In: *Proceedings of International Symposium on Graph Drawing*; 2008. p. 206–217.
- [46] Dwyer T, Koren Y, Marriott K. IPSEP-COLA: An Incremental Procedure for Separation Constraint Layout of Graphs. *IEEE Transactions on Visualization and Computer Graphics*. 2006;12(5):821–828.
- [47] Greilich M, Burch M, Diehl S. Visualizing the Evolution of Compound Digraphs with TimeArcTrees. *Computer Graphics Forum*. 2009;28(3):975–982.
- [48] Burch M, Fritz M, Beck F, Diehl S. TimeSpiderTrees: A Novel Visual Metaphor for Dynamic Compound Graphs. In: *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing*; 2010. p. 168–175.
- [49] Yi JS, Elmqvist N, Lee S. TimeMatrix: Visualizing Temporal Social Networks Using Interactive Matrix-Based Visualizations. *International Journal of Human-Computer Interaction*. 2010;26(11-12):1031–1051.

- [50] Hadlak S, Schulz HJ, Schumann H. In Situ Exploration of Large Dynamic Networks. *IEEE Transactions on Visualization and Computer Graphics*. 2011;17(12):2334–2343.
- [51] Archambault D, Purchase HC, Pinaud B. Animation, Small Multiples, and the Effect of Mental Map Preservation in Dynamic Graphs. *IEEE Transactions on Visualization and Computer Graphics*. 2011;17(4):539–552.
- [52] Feng KC, Wang C, Shen HW, Lee TY. Coherent Time-Varying Graph Drawing with Multi-Focus+Context Interaction. *IEEE Transactions on Visualization and Computer Graphics*. 2012;18(8):1330–1342.
- [53] Bach B, Pietriga E, Fekete JD. GraphDiaries: Animated Transitions and Temporal Navigation for Dynamic Networks. *IEEE Transactions on Visualization and Computer Graphics*. 2014;20(5):740–754.
- [54] Beck F, Burch M, Diehl S, Weiskopf D. The State of the Art in Visualizing Dynamic Graphs. In: *Eurographics Conference on Visualization - State of The Art Reports*; 2014. .
- [55] Singh L, Beard M, Getoor L, Blake MB. Visual Mining of Multi-Modal Social Networks at Different Abstraction Levels. In: *Proceedings of International Conference on Information Visualisation*; 2007. p. 672–679.
- [56] Ghani S, Kwon BC, Lee S, Yi JS, Elmqvist N. Visual Analytics for Multimodal Social Network Analysis: A Design Study with Social Scientists. *IEEE Transactions on Visualization and Computer Graphics*. 2013;19(12):2032–2041.
- [57] Toutanova K, Klein D, Manning C, Singer Y. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In: *Proceedings of NAACL Conference on Human Language Technology*; 2003. p. 252–259.
- [58] Gomaa WH, Fahmy AA. A Survey of Text Similarity Approaches. *International Journal of Computer Applications*. 2013;68(13):13–18.
- [59] Cilibrasi RL, Vitányi PMB. The Google Similarity Distance. *IEEE Transactions on Knowledge and Data Engineering*. 2007;19(3):370–383.
- [60] Keim DA, Mansmann F, Schneidewind J, Thomas J, Ziegler H. Visual Analytics: Scope and Challenges. In: Simoff SJ, Böhlen MH, Mazeika A, editors. *Visual Data Mining*. Springer-Verlag Berlin Heidelberg; 2008. p. 76–90.
- [61] Frey BJ, Dueck D. Clustering by Passing Messages between Data Points. *Science*. 2007;315(5814):972–976.
- [62] Shewchuk JR. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In: *Proceedings of ACM Workshop on Applied Computational Geometry*; 1996. p. 203–222.
- [63] Dörk M, Riche NH, Ramos G, Dumais S. PivotPaths: Strolling through Faceted Information Spaces. *IEEE Transactions on Visualization and Computer Graphics*. 2012;18(12):2709–2718.
- [64] Linden G, Smith B, York J. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*. 2003;7(1):76–80.

- [65] Liben-Nowell D, Kleinberg J. The Link Prediction Problem for Social Networks. In: Proceedings of International Conference on Information and Knowledge Management; 2003. p. 556–559.
- [66] Katz L. A New Status Index Derived from Sociometric Analysis. *Psychometrika*. 1953;18(1):39–43.
- [67] Crnovrsanin T, Liao I, Wu Y, Ma KL. Visual Recommendations for Network Navigation. *Computer Graphics Forum*. 2011;30(3):1081–1090.
- [68] Gu Y, Wang C, Ma J, Nemiroff RJ, Kao DL. iGraph: A Graph-Based Technique for Visual Analytics of Image and Text Collections. In: Proceedings of IS&T/SPIE Conference on Visualization and Data Analysis; 2015. .
- [69] Collins C, Penn G, Carpendale S. Bubble Sets: Revealing Set Relations with Isocontours over Existing Visualizations. *IEEE Transactions on Visualization and Computer Graphics*. 2009;15(6):1009–1016.
- [70] Dinkla K, Kreveld MV, Speckmann B, Westenberg M. Kelp Diagrams: Point Set Membership Visualization. *Computer Graphics Forum*. 2012;31(3):875–884.
- [71] Meulemans W, Riche NH, Speckmann B, Alper B, Dwyer T. KelpFusion: A Hybrid Set Visualization Technique. *IEEE Transactions on Visualization and Computer Graphics*. 2013;19(11):1846–1858.
- [72] Lorensen WE, Cline HE. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In: Proceedings of ACM SIGGRAPH Conference; 1987. p. 163–169.
- [73] Akoglu L, Faloutsos C. Event Detection in Time Series of Mobile Communication Graphs. In: Proceedings of Army Science Conference; 2010. p. 77–79.
- [74] Perron O. Zur Theorie der Matrices. *Mathematische Annalen*. 1907;64(2):248–263.
- [75] Frobenius G. Über Matrizen Aus Nicht Negativen Elementen. Walter De Gruyter Incorporated; 1912.