Similarity-Based Visualization of Large Image Collections

Chaoli Wang[a], John P. Reese[a], Huan Zhang[a], Jun Tao[a],
Yi Gu[a], Jun Ma[a], and Robert J. Nemiroff[b]

ABSTRACT

Effective techniques for organizing and visualizing large image collections are in growing demand as visual search gets increasingly popular. Targeting an online astronomy archive with thousands of images, we present our solution for image search and clustering based on the evaluation of image similarity using both visual and textual information. Time-consuming image similarity computation is accelerated using GPU. To lay out images, we introduce iMap, a treemap-based representation for visualizing and navigating image search and clustering results. iMap not only makes effective use of available display area to arrange images but also maintains stable update when images are inserted or removed during the query. We also develop an embedded visualization that integrates image tags for in-place search refinement. To show the effectiveness of our approach, we demonstrate experimental results, compare our iMap layout with a force-directed layout, and conduct a comparative user study. As a potential tool for astronomy education and outreach, we deploy our iMap to a large tiled display of nearly 50 million pixels.

Keywords: image collection, astronomy archive, image layout, embedded search, tiled display

1. INTRODUCTION

With the booming of digital cameras and image archiving and photo sharing websites, browsing and searching through large online image collections is becoming increasingly popular. An emerging trend is that images are now often tagged with names, keywords, hyperlinks and so on to improve the search and understanding. In this paper, we strive for innovation on the organization and interaction aspects of image search rather than the search algorithm itself. Specifically, we explore how to arrange images in a layout for better viewing and how to leverage the connection between image and text for better interaction.

[a]Department of Computer Science, Michigan Technological University, Houghton, MI 49931

[b]Department of Physics, Michigan Technological University, Houghton, MI 49931

Corresponding author:

Chaoli Wang, Department of Computer Science, Michigan Technological University, 1400 Townsend Drive, Houghton, MI, 49931, United States

Email: chaoliw@mtu.edu, Telephone: 1 906 487 1643

Many existing applications provide overviews of image collections by presenting a set of thumbnail images arranged in a spreadsheet-like interface. We advocate a more attractive way for image browsing to enable effective sense-making of large image collections through image ranking and clustering, and intuitive presentation and interaction. We analyze image content and design measures to evaluate their similarity using both visual and textual information. We arrange similar images close to each other and leverage a treemap-based representation to visualize image search and clustering results to facilitate the understanding. This visual means brings several benefits such as effective screen utilization, occlusion minimization, and stable update. In addition, we develop an embedded visualization that integrates image tags for in-place search refinement.

We experiment with our approach using the Astronomy Picture of the Day (APOD),[1] a popular online astronomy archive. 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* astronomical images on the Internet. This makes it perfect for us to include textual information into similarity analysis and interaction design. Our work complements the state-of-the-art image search and exploration techniques with new interface and interaction that enable effective sense-making of a large image collection. This interface guides users to sift through the collection and identify images of interest, a critical need for many applications involving with large image collections.

The paper is organized as follows. We review related work in Section 2. In Sections 3 and 4, we describe the distance computation between images, and the ranking and clustering of images. In Section 5, we discuss our iMap layout design and interaction functions. Results are given in Section 6, followed by a comparison of our iMap layout with a force-directed layout in terms of screen utilization and overlap reduction in Section 7. In Section 8, we describe the deployment of iMap to a large display wall. A user study that compares iMap with existing image search functions provided by online APOD is given in Section 9. Finally, we conclude this work in Section 10.

2. RELATED WORK

2.1 Image Similarity Analysis

Content-based image analysis is at the heart of modern image searching and retrieval. Its primary goal is to organize digital image archives by their visual content. Image features capture visual properties of an image, either globally or locally. To extract image features that help perform meaningful classification and retrieval, researchers have utilized key visual contributions such as color,[2–4] texture,[3–5] shape,[6] salient points,[7] and landmarks.[8] Advances have been made in both deriving new features and constructing signatures based on these features.[9] We leverage the color and spectrum information together with the grayscale version of images for similarity analysis.

2.2 Image Collection Organization

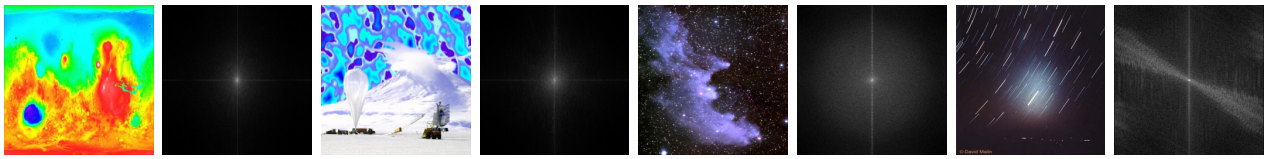
The most common way to organize a large collection of images is based on a two-dimensional grid of thumbnails, but it enforces a uniform thumbnail aspect ratio. Furthermore, only parts of the dataset can be seen within line of sight when the image collection is excessively large. Over the years, different new solutions have been proposed to improve the organization of image collections. Chen et. al.[10] leveraged the Pathfinder network scaling technique, originally developed for the analysis of proximity data in psychology, to organize a collection of images based on their color labels, texture, shape orientation etc. Torres et al.[11] introduced a focus+context approach based on spiral and concentric rings for exploring query results in an image database. Yang et al.[12] developed a scalable semantic image browser (SIB) based on the semantic content of images. The multidimensional scaling layout based on semantic similarities was used for image overview and the value and relation layout was used for content overview. Gomi et al.[13] presented clustered album thumbnails (CAT) for hierarchical browsing large image collections which shows representative images when zooming out and individual images when zooming in. Brivio et al.[14] proposed a dynamic image browsing mechanism in which the arrangement of the thumbnails is based on weighted anisotropic Voronoi diagrams and Lloyd relaxation. Tan et al.[15] designed ImageHive that generates a summary image from an image collection. A constrained graph layout algorithm was used to preserve the relationships between images and avoid occluding their salient parts, and a constrained Voronoi tessellation algorithm was applied to locally refine the layout and tile the image plane.

2.3 Visualization and Presentation Modes

Common visualization schemes for image collections include *relevance-ordered* (e.g., Google Images), *time-ordered* (e.g., the timeline[16] and time quilt[17]), *clustered* (e.g., the design gallery layout using multidimensional scaling[18]), *hierarchical* (e.g., Google Image Swirl), and *composite* (the mix of two or more of the preceding forms). In terms of user presentation, there are three modes: *static* (i.e., no motion is involved whatsoever), *moving* (i.e., constant motion even without interaction), and *interactive* (i.e., motion triggered only under user interaction). A recent study has shown that static presentation is better than moving presentation in terms of recognition success and user preference.[19] We design a layout for organizing a large image collection using the composite visualization scheme and interactive presentation mode.

3. IMAGE DISTANCE MEASURE

Measuring the similarity or distance between two images is central to any image searching or clustering tasks. Images themselves provide direct cues to visual comparison. Textual information associated with images, whenever available, gives additional hints for us to evaluate their similarity or difference. We therefore compare images



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 1. Four images and their respective grayscale frequency spectrum images. All four spectrum images are enhanced with the same log transform. We can observe that (a) and (b) are most similar, followed by (a) and (e), and (a) and (g).

using their visual and textual information from multiple perspectives and define five partial distances ($\mathbb{D}_G$, $\mathbb{D}_F$, $\mathbb{D}_H$, $\mathbb{D}_K$, and $\mathbb{D}_L$). The overall distance measure is a weighted summation of all the partial distances.

3.1 Visual Distance

Different images in an image collection come with different dimensions, types, and formats. For simplicity, we convert all resulting images to the same type and format, and scale them down to a fixed resolution (256×256). We consider three aspects of images, namely, grayscale image distance, spectrum image distance, and color histogram distance, for calculating their visual distance. Note that our solution to visual similarity analysis is by no means ideal. Rather, we seek a cost-effective solution to serve the basic visual search need.

Grayscale Image Distance. Intuitively, the similarity between two images can be evaluated by identifying their structural similarity. The structural similarity index proposed by Wang et al.[20] considers *luminance*, *contrast*, and *structure* information of the two images. We use the grayscale version of images for this evaluation and will consider color information separately when computing the color histogram distance. Given two grayscale images, we take a local 8×8 window, which moves pixel-by-pixel over the entire image, to evaluate their similarity. For two corresponding image blocks B_a and B_b, we compute their similarity as

$$\mathbb{S}_B(B_a, B_b) = \frac{(2\mu_a\mu_b + c_1)(2\sigma_{ab} + c_2)}{(\mu_a^2 + \mu_b^2 + c_1)(\sigma_a^2 + \sigma_b^2 + c_2)}, \tag{1}$$

where μ_a and μ_b are the means of B_a and B_b respectively, σ_a and σ_b are the standard deviations of B_a and B_b respectively, and σ_{ab} is the covariance of B_a and B_b. Small constants c_1 and c_2 are included to avoid instability when μ_a, μ_b, σ_a, and σ_b are very close to zero. As suggested by Wang et al.,[20] we set $c_1 = (0.01 \times L)^2$ and $c_2 = (0.03 \times L)^2$ where L is the number of levels in the grayscale images. We define the distance between two grayscale images G_a and G_b as

$$\mathbb{D}_G(G_a, G_b) = 1.0 - \frac{1}{m}\sum_{i=1}^{m} \overline{\mathbb{S}}_B(B_{ai}, B_{bi}), \tag{2}$$

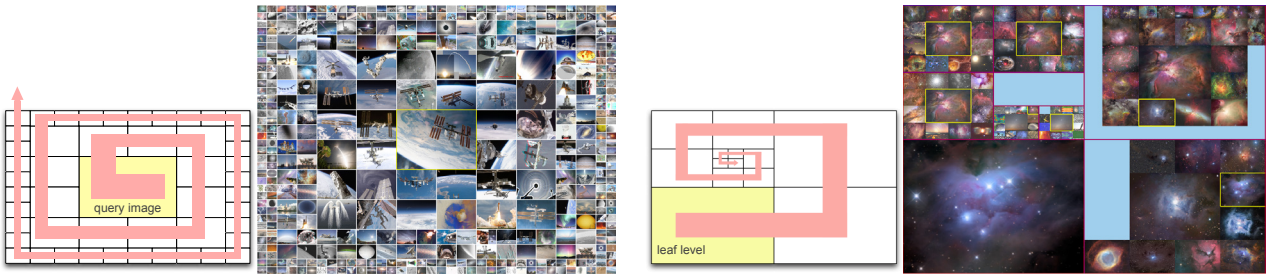


Figure 2. iMap with the spiral layout. Left: image search results. Right: hierarchical clustering results.

where m is the total number of image blocks considered and $\overline{\mathbb{S}}$ denotes the normalized similarity value.

Spectrum Image Distance. The power spectrum of an image is a representation of the magnitude of its various frequency components that has been transformed using the Fourier transform. The power at each location in the spectrum indicates the frequency and orientation of a particular feature in the image. We use the grayscale version of the spectrum image after the log transform. In Figure 1, we compare four astronomy images using their spectrum images to demonstrate the feasibility of analyzing image similarity in terms of complexity. Given two grayscale frequency spectrum images F_a and F_b, we compute their similarity through evaluating their block-wise Pearson linear correlation. Again, we take a local 8×8 window, which moves pixel-by-pixel over the entire image. For two corresponding image blocks B_a and B_b, we compute their correlation as

$$\mathbb{P}_B(B_a, B_b) = \frac{1}{n} \sum_{j=1}^{n} \left(\frac{p_{aj} - \mu_a}{\sigma_a} \right) \left(\frac{p_{bj} - \mu_b}{\sigma_b} \right), \tag{3}$$

where p_{aj} and p_{bj} are the j-th pixel values of B_a and B_b respectively, μ_a and μ_b are the means of B_a and B_b respectively, σ_a and σ_b are the standard deviations of B_a and B_b respectively, and n is the number of pixels in the block. With Equation 3, we compute the distance $\mathbb{D}_F(F_a, F_b)$ similar to Equation 2 where the *absolute* correlation values $|\mathbb{P}_B(B_{ai}, B_{bi})|$ are used instead.

Color Histogram Distance. Image colors provide additional information for similarity comparison. Given an image, we compute its color histogram by sampling each of the R, G, B channels into eight levels, which leads to a color histogram of 512 entries. Given two normalized color histograms H_a and H_b, we can use the Kullback-Leibler divergence (KLD) to evaluate their difference

$$\mathbb{K}_H(H_a || H_b) = \sum_{k=1}^{b} h_a(k) \log \frac{h_a(k)}{h_b(k)}, \tag{4}$$

where $h_a(k)$ and $h_b(k)$ are the heights of the kth bin for H_a and H_b respectively, and b is the number of bins

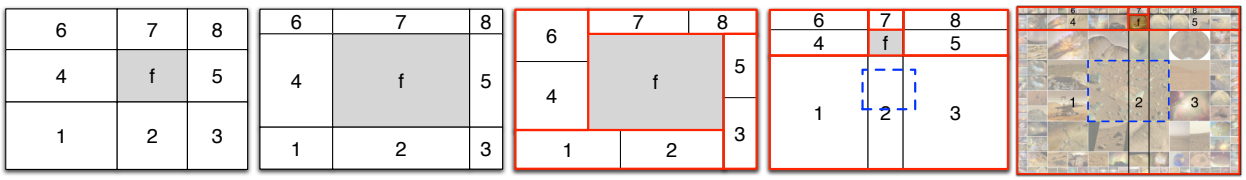


Figure 3. F+C Visualization. (a) edge expansions of the focus divide the entire space into eight areas. (b) adjust each area independently. (c) adjust areas by group. In (d) and (e), one image, shown in the dashed rectangle, crosses three areas.

in the color histogram. Notice that the KLD is not a true metric, i.e., $\mathbb{K}_H(H_a||H_b) \neq \mathbb{K}_H(H_b||H_a)$. We thus actually use the symmetric Jensen-Shannon divergence (JSD) measure instead

$$\mathbb{D}_H(H_a, H_b) = \frac{\mathbb{K}_H(H_a||H_m) + \mathbb{K}_H(H_b||H_m)}{2}, \tag{5}$$

where $H_m = (H_a + H_b)/2$.

3.2 Textual Distance

To obtain textual features, we extract meta-tagged keywords in the HTML header. We also extract hyperlinks in the explanation. These hyperlinks refer to URLs and similar hyperlinks indicate that their corresponding images may also share similarity. Then, we convert all uppercase letters to lowercase ones for extracted keywords and hyperlinks and apply the bag-of-words model[21] for textual similarity measurement. We point out that semantic-based methods for detecting text similarity[22] can classify text based on the same semantic focus such as an object or action. Nevertheless, the simple keyword extraction and similarity evaluation technique we propose performs well as all images in our case share the same theme of astronomy.

Keyword List Distance. We treat two given keywords k_a and k_b as strings and calculate their edit distance by computing the minimum number of inserts, deletes, and substitutions required to transform one string into the other. The distance between k_a and k_b is calculated as

$$\mathbb{D}_k(k_a, k_b) = \frac{c_i n_i + c_d n_d + c_s n_s}{|k_a| + |k_b|}, \tag{6}$$

where c_i, c_d, and c_s are the costs of insertion, deletion, and substitution operations respectively, n_i, n_d, n_s are the numbers of times that these three operations occur respectively, and $|k_a|$ and $|k_b|$ are the lengths of keywords k_a and k_b respectively. We set $c_i = 2.0$, $c_d = 2.0$, and $c_s = 3.0$ since the cost of deletion or insertion operation

Figure 4. Left to right: iMap showing search results with the use of visual distance only, textual distance only, and a combination of visual and textual distances, respectively.

must be greater than half of the cost of substitution operation. Otherwise, two strings k_a and k_b will be matched using the deletion and insertion operation only.

To calculate the distance between two lists of keywords, we take into account all keyword pairs from the two lists. Our solution is to calculate the average of the *minimum* distance values of keyword pairs, i.e.,

$$\mathbb{D}_K(K_a, K_b) = \frac{\mathbb{A}_K(K_a, K_b) + \mathbb{A}_K(K_b, K_a)}{2}, \tag{7}$$

where $\mathbb{A}_K(K_a, K_b)$ is the average of the minimum distances (Equation 6) for each keyword in list K_a to any keyword in list K_b. All $\mathbb{D}_\mathbb{K}$ are normalized for use.

Hyperlink List Distance. Given two hyperlinks, we compute their similarity by first determining if we are comparing an *internal* link (i.e., a URL in the same website) to an *external* link (i.e., a URL in a different website). For simplicity, we only check these links not the actual content these links refer to. If the two hyperlinks are internal and external links, we define their similarity as 0. If both links are internal, the similarity is 1 if they match exactly; otherwise, the similarity is 0. If both links are external, we ignore any directories and only take into account each hyperlink's subdomains. Each hyperlink is first split into a list, where each element in the list is a subdomain (for example, `www.nasa.gov` is split into the list `www`, `nasa`, and `gov`). We then compare each subdomain in the first hyperlink to each subdomain in the second hyperlink starting from the end of each list and moving backwards. At any step, if two subdomains being compared do not match exactly, then we do not proceed further. Let l_a be a hyperlink with $|l_a|$ subdomains, l_b be a hyperlink with $|l_b|$ subdomains, and n be the number of subdomains matched. We define the similarity between two hyperlinks l_a and l_b as

$$\mathbb{S}_l(l_a, l_b) = \frac{n}{\max(|l_a|, |l_b|)}. \tag{8}$$

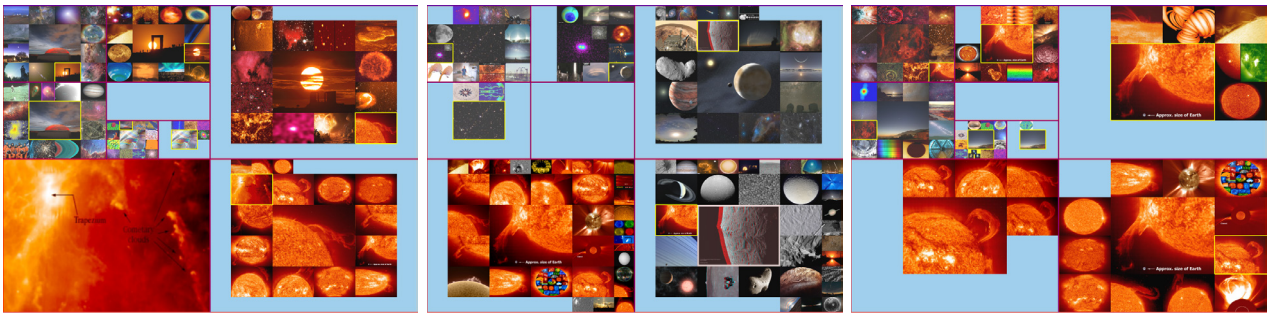


Figure 5. Left to right: iMap showing clustering results with the use of visual distance only, textual distance only, and a combination of visual and textual distances, respectively.

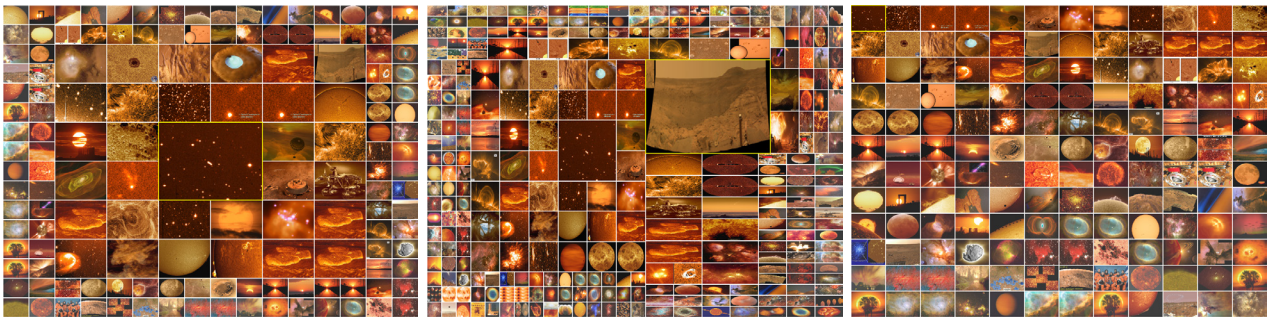
To find the similarity between two lists of hyperlinks, we use the average of the *maximum* similarity values

$$\mathbb{S}_L(L_a, L_b) = \frac{\mathbb{A}_L(L_a, L_b) + \mathbb{A}_L(L_b, L_a)}{2}, \tag{9}$$

where $\mathbb{A}_L(L_a, L_b)$ is the average of the maximum similarity values (Equation 8) for each hyperlink in list L_a to any hyperlink in list L_b. We normalize all similarity values to $[0, 1]$ and define the distance between L_a and L_b as $\mathbb{D}_L(L_a, L_b) = 1.0 - \mathbb{S}_L(L_a, L_b)$.

3.3 GPU Acceleration.

Our experiment shows that the bottleneck stages of image distance computation are grayscale image distance ($\mathbb{D}_G$) and spectrum image distance ($\mathbb{D}_F$) calculations. It is slow because we need to compute $\mathbb{S}_B$ (Equation 1) and $\mathbb{P}_B$ (Equation 3) on a local 8×8 window which moves pixel-by-pixel over the entire image. To speed up the performance, we leverage the GPU to perform distance computation in parallel. Specifically, all the thumbnail images are loaded into the GPU memory to reduce the time cost of fetching each pixel. We assign a large number of blocks and threads in the calculation for efficiency. A straightforward solution allocates N blocks and N threads within each block, where N is the number of images. In this way, each thread only needs to calculate the distance for at most one pair of images. However, this assignment has two drawbacks. First, due to the symmetry of distance matrix, we only need to calculate half of the distance matrix. This leads to an imbalanced workload for different blocks. Second, in our case, the number of images N is larger than the maximum number of threads in one block. This implies that some threads have to calculate for multiple image pairs while others are idle, which results in an imbalanced workload for different threads within a block. As a matter of fact, this straightforward solution also leads to a CUDA error `CUDA_ERROR_LAUNCH_TIMEOUT`, which occurs when a time-consuming computation runs on a graphics card that is used for both graphics rendering and CUDA computation. To avoid this problem, we evenly distribute the number of image pairs to all threads in a



(a) spiral, 5 layers, 145 images (b) spiral, 7 layers, 289 images (c) row-and-column, 144 images

Figure 6. Query by color: images are ranked according to their percentages of brown pixels. F+C visualization is shown in (b) where the focused image is expanded and highlighted in the yellow boundary.

round-robin fashion to balance the workload, and utilize a second graphics card in our PC for dedicated GPU computation.

4. IMAGE RANKING AND CLUSTERING

With the overall image distance defined, we build a symmetric distance matrix recording the distance between any two images in the collection. During image search, the user selects a query image and all other images in the collection are ranked accordingly. The user can change the weights for partial distances to update the distance matrix and search results.

For image clustering, we apply the hierarchical quality threshold algorithm due to its simplicity and efficiency. The algorithm uses a list of distance thresholds with increasing values $\{\delta_0, \delta_1, \delta_2, \ldots, \delta_l\}$ to create a hierarchy up to $l+1$ levels ($\delta_0 = 0$, $\delta_l = 1$). Initially, each image in the collection is in its own cluster. At the first iteration, we build a candidate cluster for each image I by including all images that have their respective distance to I smaller than δ_1. We save the cluster with the largest number of images as the first true cluster and remove all images in this cluster from further consideration. In the true cluster, image I is treated as its representative. We repeat with the reduced set of images until all images are classified. In the following iterations, the input is all representative images gathered from the previous iteration. We continue this process for the following iterations until we finish the lth iteration or until we only have one cluster left in the current iteration.

5. IMAGE LAYOUT AND INTERACTION

Once image are ranked, image layout is important as it determines how images should be arranged for viewing. For a large image collection, it is desirable to maintain good visual overview while allowing flexible exploration and detailed examination. An appropriate image layout should fulfill the following criteria:

Figure 7. Image search via keyword input. Left to right: the search results corresponding to keyword(s) "spiral", "galaxy", "spiral" OR "galaxy", and "spiral" AND "galaxy", respectively. There are 222, 461, 474, and 208 images matched from left to right, respectively. The first 145 images, ordered by their dates, are shown in each search.

- *Stable layout*: the layout should accommodate image ordering and maintain stable update when images are inserted or removed during the query;
- *Screen utilization*: for efficiency, the layout should display as many images as users can comfortably view them;
- *Occlusion minimization*: for effectiveness, images displayed should not occlude each other in the layout or their overlap should be minimized;
- *In-place interaction*: the layout itself should also serve as an interface for in-place interaction to attract user attention and facilitate image identification.

5.1 iMap

Since most displays and images are in the form of rectangle, we opt to use the rectangular shape for image layout so that the available display area can be best utilized. We propose to use the treemap[23] to visualize a large image collection due to its simplicity and effectiveness. We refer to the treemap-based representation of image collections as the *iMap* ("i" stands for image). Each node in iMap corresponds to a rectangle that displays an image thumbnail. The sizes of these rectangles can be determined by the importance of their images, such as search rank or hit count.

Layout Design. The original "slice-and-dice" treemap layout generates rectangles of arbitrary aspect ratios. Squarified treemaps[24] create rectangles with smaller aspect ratios but give up on node ordering. Ordered treemaps[25] offer a good tradeoff among stable updates, order preserving, and low aspect ratios. Quantum treemaps developed for PhotoMesa[26] guarantee that the regions showing groups of photos have dimensions that are integer multiples of the dimensions of the photos (i.e., they must be sized to contain quantum, or indivisible contents). Spiral treemaps[27] place nodes along the spiral pattern which guarantees that neither the overall pattern nor the specific node ordering will change drastically as the data evolve over time.

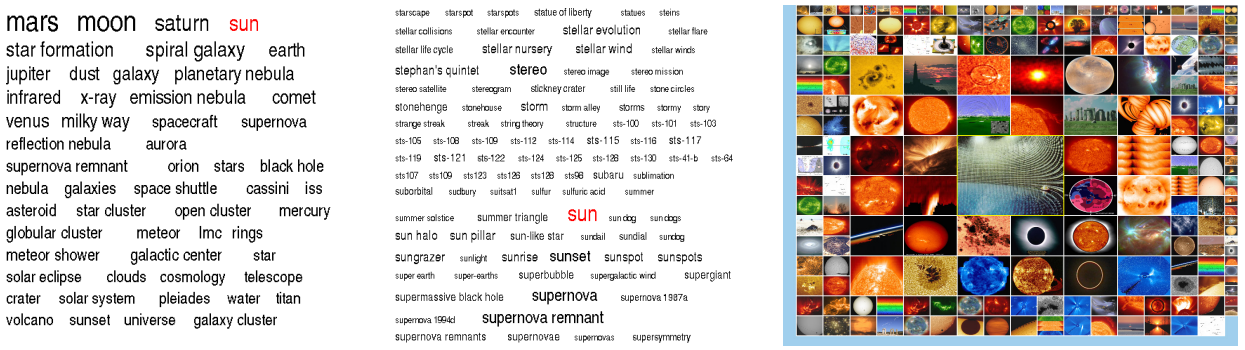


Figure 8. iMap and tag cloud. Left: the page of the most frequent keywords. Middle: a page of keywords in the alphabetical order. Right: all images that contain the keyword "sun" are ordered by their IDs and displayed.

For iMap, we propose a hybrid layout that combines the advantages of both quantum and spiral treemaps: the use of quantum with a fixed aspect ratio simplifies the layout for images of different aspect ratios; and the adoption of spiral pattern maintains stable update when we insert or remove images. The 1D spiral also accommodates image ordering, such as the chronological order or rank order. Our layout results with image search and clustering are given in Figure 2. To organize clustering results, we apply a two-level layout: the quad-tree layout for different levels of hierarchy and the spiral layout for images within each level. Much as in human vision, we display the focus image at the center of display area (focal point) and arrange less important images in its surrounding (periphery). By default, we display the focus image in normal size and reduce the width and height by half for the successive layers. The user can adjust the number of repetition layers r needed to keep the current width and height. In Figure 2, we set $r = 2$ for image search and $r = 1$ for image clustering.

iMap Interaction. The interaction with image search function includes the following: left mouse click for showing the overlay full-resolution image in its original aspect ratio; middle mouse click for showing metadata information; and right mouse click for updating search with respect to the image selected. Each image comes with an ID and we also store the number of pixels similar to a list of given representative colors. This allows us to enable "search by ID" (selecting an image ID via slider) and "search by color" (picking a representative color via radio button) functions for updating the query image besides "search by image" (directly clicking on an image displayed in iMap). The interaction with image clustering function is the same except that right mouse click on an image is for displaying its next level of images in the hierarchy.

We also implement a focus+context (F+C) function to further improve the readability of iMap as the user mouses over the images. Our F+C approach aims to apply a simple strategy to achieve an acceptable result for a single focus. As shown in Figure 3, in a general case, the expansions of the four edges of the focused image divide the entire space into eight areas. When the focus is enlarged, its edges move accordingly, so do the edges of the eight areas. However, if we simply scale each area to fit into the new layout, the four areas adjacent to the

Figure 9. iMap with embedded search. Left: image search results using visual and textual distances. Middle/right: the user clicks on the keyword "space shuttle"/"columbia" from the embedded list for result filtering.

focus, i.e., Areas 2, 4, 7, and 5, will suffer from more serious distortion than other areas (Figure 3 (b)). Taking Areas 1 and 2 for example, when the focus grows, the horizontal edges of Area 2 will expand while its vertical edges shrink; but for Area 1, both its horizontal and vertical edges will shrink. Thus, the aspect ratio of Area 2 changes more significantly than that of Area 1. To balance these changes, we group each area adjacent to the focus to the one at the corner, i.e., Areas 1 and 2, 3 and 5, 4 and 6, 7 and 8 will be grouped together, respectively. This strategy is similar to the solution proposed by Kustanowitz and Shneiderman in their two-level auto-layout technique for photo browsing.[28] Then we apply the transformation to each group, so that the overall distortion can be reduced (Figure 3 (c)). However, for the spiral layout, an image that is larger than the focus might cross three areas, as shown in Figure 3 (d). Figure 3 (e) shows an example where the current query image at the center crosses Areas 1, 2, and 3. In this case, Areas 1, 2, and 3 must be in one group to ensure that every image is still a rectangle. For other areas, we will group Areas 4 and 6, 5 and 8, respectively, and leave Area 7 ungrouped (since it is the smallest one).

In our implementation, the focused image will be scaled up to the same size as the centered query image before the deformation. To better preserve the original aspect ratio for the areas out of focus, the center of the focused image might move. Take the horizontal direction for example, the left and right boundaries of the focused image after deformation are decided in a way such that the left and right remaining areas are squeezed proportionally to their original widths. Once the boundaries of the focused image are determined, we compute the boundaries of each area and uniformly deform the context images within each area.

5.2 Integrating Text into Image Search

Building the overall distance matrix that includes the keyword and hyperlink distances for image search *implicitly* utilizes the textual information. In this case, all keywords or all hyperlinks associated with images must be taken into account. Nevertheless, these keywords or hyperlinks can also be *explicitly* utilized to customize or refine the search. We present three different ways to explicitly integrate text into image search.

			CPU			GPU				
similarity type	image number	step size	loading time (s)	computing time (s)	saving time (s)	block number	thread number	loading time (s)	computing time (s)	saving time (s)
grayscale	4560	1×1	1.91	976,200	1.04	16384	512	0.2194	53,886	0.0338
		2×2						0.2906	12,162	0.0098
		4×4						0.215	3,015	0.0077
		8×8						0.214	762	0.0098
spectrum	4560	1×1	1.52	787,149	2.43	16384	1024	0.2158	31,944	0.0077
		2×2						0.222	7,993	0.0071
		4×4						0.2138	2,017	0.0077
		8×8						0.2136	515	0.0015
histogram	4560	N/A	1.08	176.52	2.6	16384	1024	0.0021	3.2596	0.0086

Table 1. Parameter values and timing results for grayscale image distance, spectrum image distance, and color histogram distance calculations. CPU loading and saving are loading image data from disk and saving the distance matrix to disk. GPU loading and saving are copying the data from main memory and saving the distance matrix to main memory. For GPU computation of grayscale image distance and spectrum image distance, we also report timing with different step sizes for the local 8×8 window.

Keyword Input. In this mode, the user inputs a keyword into a text widget and we search and display images that contain such a keyword. The keyword input could be partially or exactly matched with image keywords and/or text explanations in the HTML files. Multiple keywords are allowed with logical operators (OR, AND). Images founded can be ordered by their IDs or other attributes such as hit count. They can also be arranged according to their rank order from the previous image search results.

Tag Cloud. Unlike keyword input, tag cloud displays pages of keywords from which the user clicks on a keyword of interest to find related images. The order of keywords in the tag cloud can be determined by their alphabetical or frequency order. Their visual attributes such as size can be determined by their frequency.

Embedded Text. Both keyword input and tag cloud display text in another widget or window separate from iMap. Another different design is to embed image tags for *in-place* selection and search, which obviates the need to use a list of words separately. This concept is similar to PhotoMesa,[26] a zoomable image browser (in terms of in-place interaction) and scented widgets[29] (in terms of improving navigation cues). Zoomable interfaces make navigation straightforward and avoid getting lost. Improving navigation cues lowers the cost structure of seeking and accessing information. We therefore advocate a solution that integrates tag information into iMap for further interaction and embedded search. For the APOD collection, tags are only associated with the image, not specific objects. We thus overlay a layer of tags for user selection and search refinement. Much as in tag clouds, the size, color, or order of tags for an image can be adjusted to provide additional hints such as how many images will match selected tags. The user can select multiple tags to add into the current search.

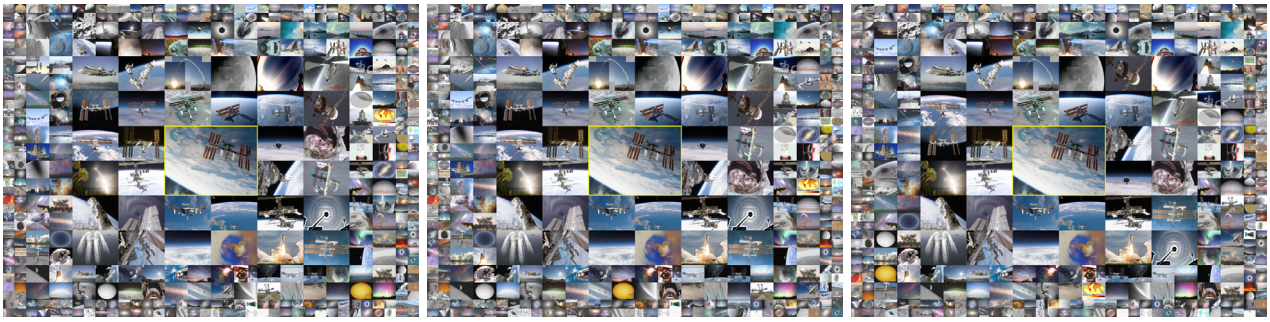


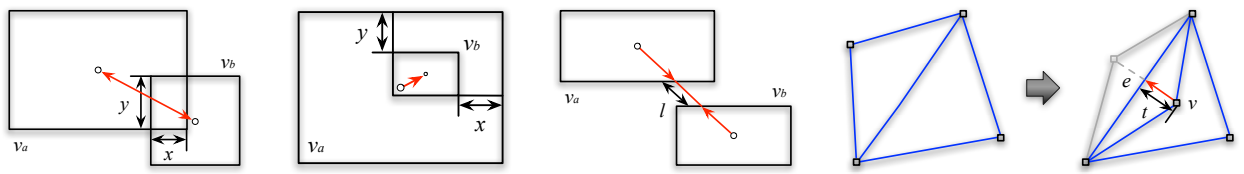
Figure 10. Comparing the search results with different step sizes for the local 8×8 window used in computing grayscale image distance and spectrum image distance. The ground truth search results are shown in Figure 2. From left to right: search results with step size of 2×2, 4×4, and 8×8, respectively.

6. RESULTS

We collected APOD webpages that contain meta-tagged keywords (since Sep. 26, 1997) till a cut-off date (Apr. 3, 2010). Occasionally, APOD runs videos instead of images. In this experiment, we did not consider videos and therefore excluded those webpages from our collection. For images in the GIF format though, we extract the first frame as the representative of the entire image sequence. The resulting data set consists of 4560 images with text information including keywords, hyperlinks, and explanations extracted from the HTML files. At preprocessing, we computed five distance matrices ($\mathbb{D}_G$, $\mathbb{D}_F$, $\mathbb{D}_H$, $\mathbb{D}_K$, and $\mathbb{D}_L$) to record the distance for all pairs of images. These distance matrices are used at runtime to update the final overall distance. In the following, we present iMap results with screenshots captured from our program.

6.1 Image Layout

Figure 2 shows iMap layout. For image search, the query image of the International Space Station is displayed at the center of iMap as the focus. Results based on both visual and textual distances are ranked and arranged along the spiral circling out. The effectiveness of image search can be verified by the similar images retrieved and displayed. For image clustering, the user explores the cluster hierarchy by clicking the image of interest which will be highlighted with a yellow bounding box, and its next level of detail is displayed. By displaying all levels of hierarchy currently explored, we give the user the freedom to jump between non-consecutive hierarchical levels during the exploration. In Figure 4, we can see that using only visual distance picks up images of similar brownish-yellow colors while using only textual distance pulls up the ranks for grayscale images related to the Mars. A combination of both visual and textual distances finds a balance in between. In Figure 5, we show that if no other textual search is provided, clustering images with the incorporation of visual distance would be more effective for users to identify images of interest in their exploration. Our current implementation only provides the top-down exploration of the hierarchy, it would be interesting to consider the bottom-up exploration



(a) (b) (c) (d)

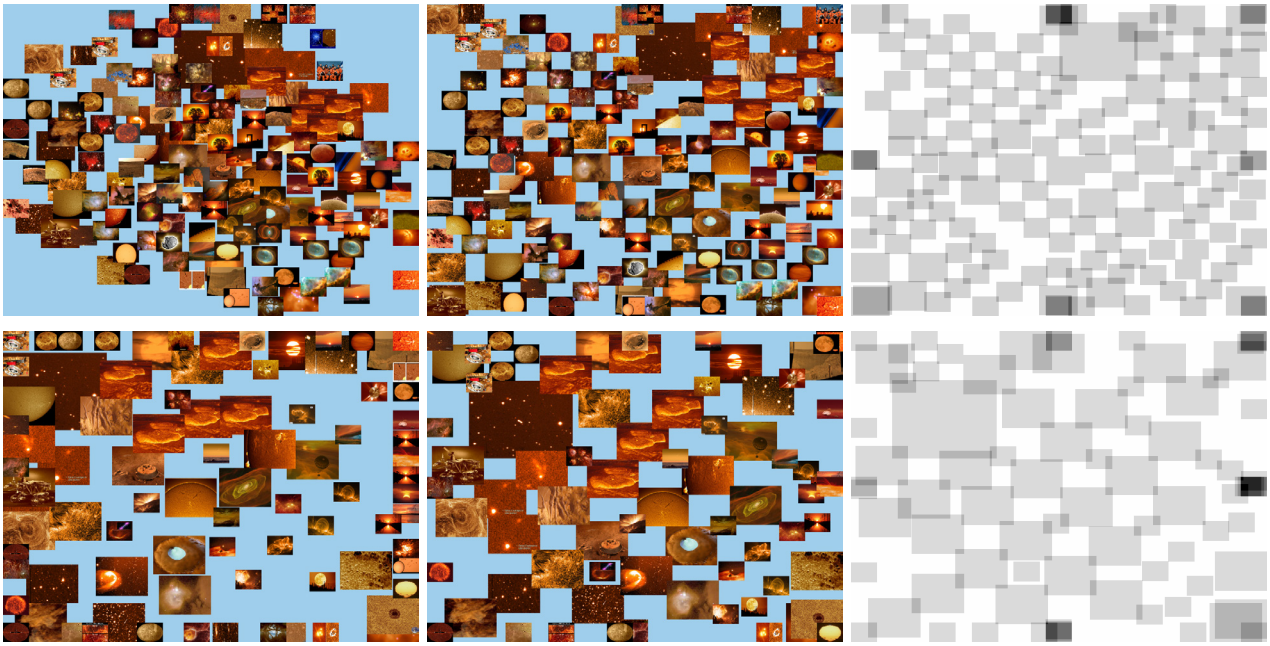
Figure 11. The four different forces we consider for the force-directed layout adjustment for overlap reduction and topology maintenance: (a) bidirectional repulsive force, (b) unidirectional repulsive force, (c) spring force, and (d) attractive force.

as well. For example, the user can search a leaf image via keyword and all corresponding intermediate levels in the hierarchy are automatically extracted and displayed.

Figure 6 shows an example of "search by color" where images are ranked according to how much percentages of brown pixels they contain. Compared to the row-and-column layout, the spiral layout effectively highlights top-ranked images while maintaining stable update when the number of image layers changes. We also show the results with different numbers of layers displayed and the effect with F+C visualization. Note that our F+C strategy might result in larger distortion when the focused image is close to the corner. In this case, some divided areas could be small, whose size will change dramatically during the deformation. A more sophisticated F+C visualization for this kind of application remains an open problem. To keep the shape of each image as a rectangle, it will be challenging to minimize the distortion while preserving their relative positions without creating voids. The variation of image size in the spiral layout makes the problem more complicated, since those large images will greatly limit the possible moves we can take. Producing a smooth animated transition of the deformation could be even more difficult.

6.2 Image and Text

Figure 7 shows an example of interactive image filtering via keyword input. The user can choose either partial or exact keyword match in the search. Figure 8 shows the use of tag cloud in iMap. Tag clouds organize all keywords in a certain order and the user can go over pages to identify the keyword of interest. As an option, the frequency count for each keyword may also be displayed. When the user clicks on the keyword, iMap updates the search result and displays all images containing the selected keyword. Figure 9 shows an example where the user first searches images based on the use of visual and textual distances as usual. Upon identifying the images of interest, the user can then turn on the embedded keyword list associated with the query image to refine the search by clicking on a certain keyword. All images that do not contain the selected keyword will be filtered out from the query result while the rank order from the previous search is utilized to maintain the relative stable update. The embedded search allows users to perform in-place interaction to refine their search results without shifting their focus among different windows, which makes it easy for users to follow and take actions.



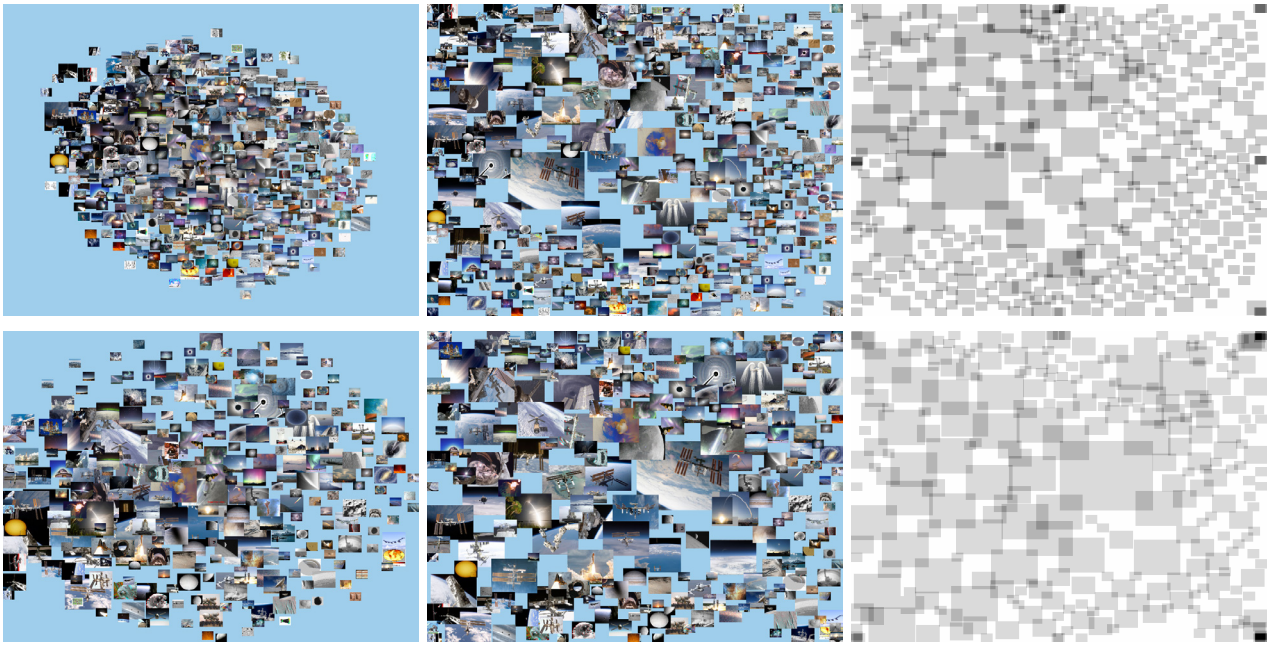
(a) (b) (c)

Figure 12. Force-directed layouts based on the iMap result shown in Figure 6 (a). (a) the initial force-direct layout. (b) the force-directed layout after overlap reduction. (c) quad drawing showing the actual overlap in (b). First row: scaling down the images so that the summation of all image sizes accounts for 75% of the display area as the input. $o = 14.00\%$, $u = 66.39\%$. Second row: removing the last layer of images from the iMap result as the input. $o = 20.46\%$, $u = 65.00\%$.

6.3 Performance

For the APOD data set, the one-time computation of the five distance matrices took days to complete on a single Intel 3.2GHz CPU. The dominant timing was spent on calculating $\mathbb{D}_G$ and $\mathbb{D}_F$ where we computed $\mathbb{S}_B$ (Equation 1) and $\mathbb{P}_B$ (Equation 3) on a local 8×8 window which moves pixel-by-pixel over the entire image. Leveraging an nVIDIA GeForce GTX 580 graphics card to perform distance computation in parallel for multiple images, we are able to reduce the computation of the three image distance matrices significantly. The configurations and timing results are shown in Table 1. We can see that GPU computation provides about 18X, 24X, and 54X speedup for grayscale image distance, spectrum image distance, and color histogram distance calculations, respectively. The reason that we only allocate 512 threads in the calculation of grayscale image distance is due to its complexity. It requires more local memory and registers than the other two, thus using more threads causes the crash of GPU.

This timing performance can be further improved by taking an approximate solution: setting a step size larger than one pixel for moving the local window. Table 1 shows the timing with three approximate solutions (with step size of 2×2, 4×4, and 8×8, respectively). The speedup is nearly linear to the reduction rate (i.e.,



(a) (b) (c)

Figure 13. Force-directed layouts based on the iMap result shown in the left image of Figure 2. (a) the initial force-direct layout. (b) the force-directed layout after overlap reduction. (c) quad drawing showing the actual overlap in (b). First row: scaling down the images so that the summation of all image sizes accounts for 75% of the display area as the input. $o = 11.08\%$, $u = 68.46\%$. Second row: removing the last layer of images from the iMap result as the input. $o = 19.84\%$, $u = 75.45\%$.

4X, 16X, and 64X for 2×2, 4×4, and 8×8, respectively). When using the step size of 8×8 (in this case, the local sliding window does not overlap each other), the total GPU computation time for generating the three image distance matrices is reduced to less than 30 minutes, which is quite affordable. In Figure 10, we show that these approximate solutions do not lead to dramatic changes of image search results. All five distances (three image distances and two textual distances) take an equal weight in the final distance measure. We can see that there are changes of ranking for the similar images with respect to the query image as we adjust the step size for the local window, but compared with the ground truth, the overall approximate results are fairly good even with a large step size of 8×8. In fact, for results with these three step sizes shown in Figure 10, 99.76% (2×2), 99.52% (4×4), and 99.04% (8×8) of images are the same as the images shown in Figure 2. At run time, only the clustering step takes a few minutes to complete. All other tasks and interactions are interactive.

7. COMPARISON WITH FORCE-DIRECTED LAYOUT

We compare our iMap layout with a force-directed layout in terms of screen utilization and overlap reduction. We apply the Fruchterman-Reingold algorithm[30] to draw an initial force-directed graph layout due to its popularity

and simplicity. For the initial layout, we use images retrieved from the same query using iMap as the input for nodes. As we know, iMap produces no image overlapping and could fully utilize the display area. This would not be possible for the force-directed layout. For meaningful comparisons, we have to either reduce the number of images displayed, or proportionally scale down the size of each image (the images with the smallest size may remain unchanged for easy readability). To reduce image overlap while maintaining the relative positions among different images, we apply a triangulation scheme[31] to the initial layout and use the resulting mesh to perform constrained layout adjustment. Similar to the algorithms described by Cui et al.,[32] we consider four kinds of forces to reposition the nodes to reduce their overlap while maintaining the topology of the layout:

- *Bidirectional repulsive force:* This force pushes away two nodes v_a and v_b from each other and is effective if and only if v_a and v_b overlap each other. The bidirectional repulsive force 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 region as shown in Figure 11 (a). This force is applied to every pair of nodes in the layout.

- *Unidirectional repulsive force:* This force pushes away a node v_b from a node v_a and is effective if and only if v_b is inside v_a. The unidirectional repulsive force 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 as shown in Figure 11 (b). Once this force pushes v_b away from v_a and they become overlapping, the previous bidirectional repulsive force will be effective in subsequent iterations to move them further apart.

- *Spring force:* This force is used to balance 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 as shown in Figure 11 (c). This force is applied to every pair of nodes in the graph.

- *Attractive force:* This force is used to maintain the underlying triangle mesh we construct for the graph. During the layout adjustment, if a mesh triangle is flipped, as shown in Figure 11 (d), then the topology of the triangle mesh changes. Our goal is to maintain a 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 four corners of the drawing area. This is to ensure that all graph nodes do not go out of bound.

Figures 12 and 13 show two examples of the force-direct layouts based on Figure 6 (a) (five layers, 145 images) and the left image of Figure 2 (seven layers, 417 images), respectively. We compute the amount of image overlap (in pixels) and derive the percentage of image overlapping as follows

Figure 14. iMap shown on the 24-monitor tiled display that can display nearly 50 million pixels simultaneously.

$$o = \frac{\sum_{i \in N, j \in N, i<j} A_i \cap A_j}{A}, \tag{10}$$

where $A_i \cap A_j$ is the number of pixels overlapped by images i and j, N is the total number of images in the layout, and A is the number of pixels in the entire display area. The percentage of screen utilization is computed as $u = \bar{A}/A$, where $\bar{A}$ is the number of pixels that are occupied by at least one image in the layout.

In Figure 12, we either reduce the number of images from 145 to 85 images by removing the outmost layer of the iMap result, or proportionally scale down the size of each image (except the last two layers) so that the summation of all image sizes accounts for 75% of the display area. After applying overlap reduction, the force-directed layout still has 14 to 20% image overlap with about one third of pixels not utilized. These numbers improve slightly with seven layers of images in Figure 13. As a comparison, the iMap layout has zero image overlap while all pixels are utilized. Besides this major advantage, we find that the iMap layout is clearer as we only consider the similarity between the query image and each of the other images in the layout. In contrast, the force-directed layout considers the similarity between all pairs of images in the layout.

8. DEPLOYMENT OF IMAP TO TILED DISPLAY

To transform iMap into a tool for astronomy education, we deploy our system to a large tiled display at Michigan Technological University's Immersive Visualization Studio. The tiled display consists of 6×4 thin-bezel 46-inch Samsung monitors, each with 1920×1080 pixels. These 24 monitors are driven by eight computing nodes for computation and visualization and each node connects to three monitors. In total, the tiled display can display nearly 50 million pixels simultaneously. In order to forward the iMap window from the local desktop's monitor to the tiled display wall, we leverage the open-source libraries Chromium,[33] a system for interactive rendering on clusters of workstations. Chromium provides a number of key features for large tiled display wall rendering. For example, it can synchronize parallel graphics commands to implement various parallel rendering techniques.

It can also aggregate the output of multiple graphics cards to form a single display with higher performance. Furthermore, since Chromium streams the graphics pipeline based on the industry standard OpenGL API, it is transparent to the programmers and allows many OpenGL programs to run without modification. By specifying the hardware configuration into the Chromium config file, users can easily set up Chromium and fit it into their own tiled display architecture.

Some photo results showing the running of iMap on the tiled display are shown in Figure 14. Currently, we are using this facility for showing iMap demos to visitors and teaching middle and high school students through Michigan Tech's Youth Programs. Initial feedback from several groups of visitor is fairly positive as they comment that running iMap on this life-size tiled display is much more expressive and fun to watch compared with 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 small images close to the wall's boundary.

9. USER STUDY

We performed a user study to evaluate the effectiveness of iMap by comparing it with the existing image search functions (archive, index, and text search) provided by online APOD. A desktop version of iMap was used in the study, running on a 27-inch monitor with 1920×1080 screen resolution. We used a design of 2 conditions (iMap vs. online APOD) $\times$ 3 tasks (text, image, and image + text). We assigned a target image for each of the six combinations except for Task 1, where users were asked to identify two images with very different numbers of images retrieved. So, a total of eight images were selected. These images cover different topics: astronaut, aurora, black hole, Earth, Jupiter, Mars, Moon, and Sun.

9.1 Hypotheses

We postulated four hypotheses for the study. Since the users' respond time varies for each task, hypotheses about response time of search under both conditions will be considered based on different tasks. Furthermore, we only considered the overall accuracy due to the high probability of finding the exact image.

- *Hypothesis 1.* Given the keywords and description only (Task 1), iMap is faster to search than online APOD.
- *Hypothesis 2.* Given the image only (Task 2), iMap is faster to search than online APOD.
- *Hypothesis 3.* Given the image and keywords (Task 3), iMap is faster to search than online APOD.
- *Hypothesis 4.* Overall, for image search, using iMap makes fewer errors than online APOD.

9.2 Interactions

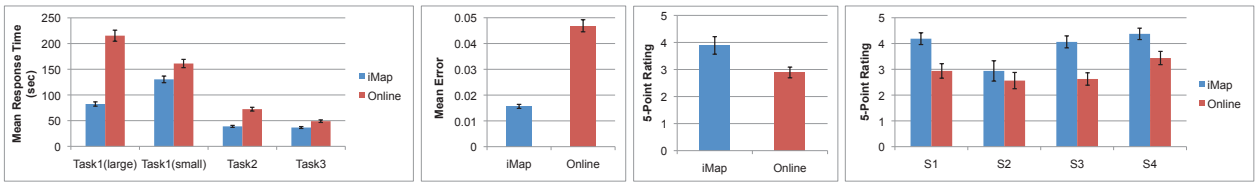
Detailed interactions with iMap are described in Section 5. For online APOD, three search modes are provided: archive, index, and text search. The archive mode provides the dates and titles of all images and arranges them in the reverse chronological order. The index mode offers various keywords organized by category. Clicking on any keyword shows keyword-related thumbnail images with their dates, titles, and short explanations. The text search mode provides OR or AND search for multiple keywords. Since the way to search online APOD is similar to other websites, we assume that all users are familiar with it.

9.3 Tasks and Procedure

Three tasks were implemented to compare the performance of iMap and online APOD. In each task, users were asked to identify the three most related or similar images. Ideally, the exact image should be found and if not, they were asked to find up to three most related ones.

- *Task 1.* Text search: given a short text description and several recommended keywords, users were asked to find the three most related images. Two images will be tested for each condition, one search generated a large number of retrieved images while the other generated a small number.
- *Task 2.* Image search: given target images without any keyword or description, users were asked to find the three most similar images. We assumed that the users could figure out the content of the images with their very basic astronomy knowledge.
- *Task 3.* Image + text search: given target images with several keywords, users were asked to find the three most similar images.

For iMap, there was a 15-minute training session and a further 5-minute free exploration time (practice search) preceding the actual tasks. This was the same for online APOD, except that there was no training. A post-test survey for user preference and comments was conducted immediately after a user finished all the tests. A total of 16 users (eight graduate and eight undergraduate students) participated in this study and each user used both iMap and online APOD. Users were required to finish tasks in the order given. Users recorded the date of images they selected, and we helped them record the starting and ending time for completing each task. Each user was asked to perform 8 trials, and therefore, we had a total of 128 (16×8) trials. Each experiment was conducted individually and took approximately 40 minutes, including the training, practice task, experimental tasks for both conditions, and questionnaire.



(a) (b) (c) (d)

Figure 15. Comparing iMap to online APOD. (a) mean response time by task. (b) mean error rate. (c) mean overall rating. (d) mean rating by statement. The four statements are: (S1) It was enjoyable to use; (S2) It was easy to search images with only keywords given; (S3) It was easy to search images with only images given; and (S4) It was easy to search images with keywords and images given.

9.4 Results and Discussion

We present results from the study in three aspects: completion time, accuracy, and subjective preferences. A paired t-test with a standard significance level $\alpha = 0.05$ was performed to determine statistical significance between the two conditions.

Completion Time. Figure 15 (a) shows the mean response time results when comparing iMap to online APOD for each task.

- For Task 1, iMap was significantly faster than online APOD with only keywords given and a large number of images retrieved (iMap: 82.39s, online APOD: 215.36s; $p = 0.007$). This is because iMap provides a good overview of the image collection to facilitate the visual search. However, no significant difference was found with only keywords given and a small number of images retrieved. Possible reasons include: the text descriptions were confusing to some users; users were more proficient for online search while it took time for them to adjust to using iMap; and users needed to switch among multiple views or tabs in iMap while there was only one view in online APOD.

- For Task 2, iMap was significantly faster than online APOD with only images given (iMap: 38.86s, online APOD: 72.29s; $p = 0.0127$). Two reasons explain this. First, using online APOD, users were able to scroll up and down the result page to view about ten images at a time, while they could view hundreds of image simultaneously using iMap. Second, neighboring images in iMap are similar while neighboring images have no connection at online APOD.

- For Task 3, no significant difference in response time was found with keywords and images given. Since the images we selected for this task got a very small number of images retrieved (10 to 20 images), it was easy for users to search under both conditions.

Therefore, Hypothesis 1 was fully supported when the number of images retrieved is large. Hypothesis 2 was also fully supported while Hypotheses 3 was not supported.

Accuracy. Since almost all users were able to find the exact image correctly, we compute the error as the number of failed trials (i.e., image misidentified) over the number of total trials. With online APOD, 4.69% (6 out of 128) tests failed and with iMap, 1.56% (2 out of 128) tests failed. Figure 15 (b) shows the average error results when comparing iMap and online APOD. In terms of accuracy, iMap was not distinguishable from online APOD, which contradicts Hypothesis 4. We found that almost every user could find the exact image under both conditions.

Subjective Preferences. The 16 users completed a survey after their experiment. They were asked which condition they preferred overall and which interface they perceived to be more useful for each of the three tasks. Four statements were provided as follows

- *Statement 1.* "It was enjoyable to use."
- *Statement 2.* "It was easy to search images with only keywords given."
- *Statement 3.* "It was easy to search images with only images given."
- *Statement 4.* "It was easy to search images with keywords and images given."

Each statement was answered with a 5-point scale (1 = strongly disagree, 5 = strongly agree). We ran a paired t-test with a standard significance level $\alpha = 0.05$ and found a significant effect for iMap.

Figure 15 (c) shows the mean overall rating when comparing iMap to online APOD. The rating for iMap is significantly higher than online APOD (iMap: 3.89, Online APOD: 2.89; $p = 0.039$), which indicates that the users prefer using iMap over online APOD.

Figure 15 (d) shows the mean rating when comparing iMap to online APOD for the four statements:

- For Statement 1, iMap was judged to be more enjoyable to use (iMap: 4.19, online APOD: 2.94; $p = 0.0017$).
- For Statement 2, iMap was not significantly easier to search images with only keywords given (iMap: 2.94, online APOD: 2.56; $p = 0.5544$).
- For Statement 3, iMap was significantly easier to search images with only images given (iMap: 4.06, online APOD: 2.634; $p = 0.0001$).
- For Statement 4, iMap was significantly easier to search images with keywords and images given (iMap: 4.38, online APOD: 3.44; $p = 0.0098$).

10. CONCLUSIONS AND FUTURE WORK

We have presented iMap, an analysis and visualization framework that supports effective searching, browsing, and understanding of large image collections. iMap strikes a good balance among simplicity, intuitiveness, and effectiveness by addressing issues such as stable layout, screen utilization, and in-place interaction. Our user study confirms that iMap provides a more effective solution for image search, ranking, and identification compared with traditional archive and keyword search methods. While the layout itself will work for other image collections beyond APOD, we would improve our image distance measure by considering image aesthetics measures, high-level image features, and relationships between words so that the similarity ranking would be applicable to other image collections as well. In the future, we will also develop a web version of iMap for APOD so that any users can easily access our system online.

ACKNOWLEDGMENTS

This work was supported in part by the U.S. National Science Foundation [IIS-1017935, CNS-1229297]. 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] Swain M, Ballard B. Color Indexing. International Journal of Computer Vision. 1991;7(1):11–32.

[3] Rodden K, Basalaj W, Sinclair D, Wood K. Evaluating a Visualisation of Image Similarity as a Tool for Image Browsing. In: Proceedings of IEEE Symposium on Information Visualization; 1999. p. 36–43.

[4] Rodden K, Basalaj W, Sinclair D, Wood K. Does Organisation by Similarity Assist Image Browsing? In: Proceedings of ACM SIGCHI Conference; 2001. p. 190–197.

[5] Manjunath BS, Ma WY. Texture Features for Browsing and Retrieval of Image Data. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1996;18(8):837–842.

[6] Bimbo AD, Pala P. Visual Image Retrieval by Elastic Matching of User Sketches. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1997;19(2):121–132.

[7] Mikolajczyk K, Schmid C. Scale and Affine Invariant Interest Point Detectors. International Journal of Computer Vision. 2004;60(1):63–86.

[8] Kennedy L, Naaman M. Generating Diverse and Representative Image Search Results for Landmarks. In: Proceedings of International Conference on World Wide Web; 2008. p. 297–306.

[9] Datta R, Joshi D, Li J, Wang JZ. Image Retrieval: Ideas, Influences, and Trends of the New Age. ACM Computing Surveys. 2008;40(2).

[10] Chen C, Gagaudakis G, Rosin P. Content-Based Image Visualization. In: Proceedings of International Conference on Information Visualisation; 2000. p. 13–18.

[11] 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.

[12] 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.

[13] Gomi A, Miyazaki R, Itoh T, Li J. CAT: A Hierarchical Image Browser Using a Rectangle Packing Technique. In: Proceedings of International Conference on Information Visualisation; 2008. p. 82–87.

[14] 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.

[15] Tan L, Song Y, Liu S, Xie L. ImageHive: Interactive Content-Aware Image Summarization. IEEE Computer Graphics and Applications. 2012;32(1):46–55.

[16] Harada S, Naaman M, Song YJ, Wang QY, Paepcke A. Lost in Memories: Interacting With Large Photo Collections on PDAs. In: Proceedings of ACM/IEEE-CS Joint Conference on Digital Libraries; 2004. p. 325–333.

[17] Huynh DF, Drucker SM, Baudisch P, Wong C. Time Quilt: Scaling up Zoomable Photo Browsers for Large, Unstructured Photo Collections. In: Proceedings of ACM CHI Extended Abstracts; 2005. p. 1937–1940.

[18] Marks J, Andalman B, Beardsley PA, Freeman W, Gibson S, Hodgins J, et al. Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. In: Proceedings of ACM SIGGRAPH Conference; 1997. p. 389–400.

[19] Cooper K, de Bruijn O, Spence R, Witkowski M. A Comparison of Static and Moving Presentation Modes for Image Collections. In: Proceedings of International Working Conference on Advanced Visual Interfaces; 2006. p. 381–388.

[20] Wang Z, Bovik AC, Sheikh HR, Simoncelli EP. Image Quality Assessment: From Error Visibility to Structural Similarity. IEEE Transactions on Image Processing. 2004;13(4):600–612.

[21] Lewis D. Naïve (Bayes) at Forty: The Independence Assumption in Information Retrieval. In: Proceedings of European Conference on Machine Learning; 1998. p. 4–15.

[22] Hatzivassiloglou V, Klavans JL, Eskin E. Detecting Text Similarity over Short Passages: Exploring Linguistic Feature Combinations via Machine Learning. In: Proceedings of Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora; 1999. p. 203–212.

[23] Shneiderman B. Tree Visualization with Tree-Maps: A 2D Space-Filling Approach. ACM Transactions on Graphics. 1992;11(1):92–99.

[24] Bruls M, Huizing K, van Wijk JJ. Squarified Treemaps. In: Eurographics/IEEE TCVG Symposium on Visualization; 1999. p. 33–42.

[25] Shneiderman B, Wattenberg M. Ordered Treemap Layouts. In: IEEE Symposium on Information Visualization; 2001. p. 73–78.

[26] Bederson BB. PhotoMesa: A Zoomable Image Browser Using Quantum Treemaps and Bubblemaps. In: Proceedings of ACM Symposium on User Interface and Software Technology; 2001. p. 71–80.

[27] Tu Y, Shen HW. Visualizing Changes of Hierarchical Data Using Treemaps. IEEE Transactions on Visualization and Computer Graphics. 2007;13(6):1286–1293.

[28] Kustanowitz J, Shneiderman B. Hierarchical Layouts for Photo Libraries. IEEE Multimedia. 2006;13(4):62–72.

[29] Willett W, Heer J, Agrawala M. Scented Widgets: Improving Navigation Cues with Embedded Visualizations. IEEE Transactions on Visualization and Computer Graphics. 2007;13(6):1129–1136.

[30] Fruchterman TMJ, Reingold EM. Graph Drawing by Force-Directed Placement. Software - Practice and Experience. 1991;21(11):1129–1164.

[31] 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.

[32] 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.

[33] Chromium;. `http://chromium.sourceforge.net/`.