MediaGLOW: Organizing Photos in a Graph-based Workspace

Andreas Girgensohn[1], Frank Shipman[2], Lynn Wilcox[1], Thea Turner[1], Matthew Cooper[1]

[1]FX Palo Alto Laboratory, Inc.
3400 Hillview Avenue
Palo Alto, CA 94304, USA

[2]Department of Computer Science &
Center for the Study of Digital Libraries
Texas A&M University
College Station, TX 77843-3112

{andreasg, wilcox, turner, cooper}@fxpal.com, shipman@cs.tamu.edu

ABSTRACT

We designed an interactive visual workspace, MediaGLOW, that supports users in organizing personal and shared photo collections. The system interactively places photos with a spring layout algorithm using similarity measures based on visual, temporal, and geographic features. These similarity measures are also used for the retrieval of additional photos. Unlike traditional spring-based algorithms, our approach provides users with several means to adapt the layout to their tasks. Users can group photos in stacks that in turn attract neighborhoods of similar photos. Neighborhoods partition the workspace by severing connections outside the neighborhood. By placing photos into the same stack, users can express a desired organization that the system can use to learn a neighborhood-specific combination of distances.

ACM Classification Keywords: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems – *photos*.

Keywords: Photo organization, retrieval, sharing.

INTRODUCTION

Today, it is fairly common for users to have large photo collections (e.g., Flickr or personal photos) and to sift through them to create specialized groupings for sharing (e.g., scrapbooks) or presentation (e.g., special occasions). Such tasks require the selection of appropriate photos, the creation of categories, the attachment of labels to organize a collection, and the interactive visualization of collections. Many photo applications provide hierarchic and grid-based visualizations that limit user expression for such activities.

We created MediaGLOW (Graph Layout Organization Workspace) to explore a different approach to supporting users in locating and organizing photos (see Figure 1). Its innovation lies in a workspace that integrates a variety of visualization and interaction techniques with image classifiers, enabling users to locate, collect, and tag their photo collections for sharing or presentation. Image classifiers allow users to see their photo collections grouped by different similarity measures. A graph layout mechanism indicates this similarity visually by placing graph nodes (i.e., photos) in a 2D space while minimizing differences between desired and actual distances between photos. The visual similarity measure takes advantage of manually tagged data available from the Flickr photo sharing site. Classifiers trained on tagged photos predict the likelihood for certain tags that in turn generates the visual inter-photo distance measure.

When photos are dragged to different positions in the workspace, related photos follow according to similarity. Users may group photos into stacks to organize the workspace to facilitate their current activity. The system changes its understanding of the user-created categories as photos are moved in to, out of, and between stacks. The photo stack is thus a visual organizational metaphor and is represented by a single node in the layout graph. Stacks have neighborhoods of highly similar photos that are visualized by a halo. They pull additional related photos into their neighborhoods for further consideration by the user. Stack surrogates in a side bar simplify the location and manipulation of stacks. Attribute lists and trees support the selection and retrieval of matching photos. These features go beyond traditional photo organization systems that support the users in browsing photos temporally or geographically and also extend existing image graph layout systems.

RELATED WORK

Many applications for organizing photos present a list or grid of thumbnails, e.g. the image search results for Google and Flickr and the browsing interface for Picasa. Several applications group photos semi-automatically (e.g., [8]) while others use graph-based layouts for presenting images.

Visual Who [2] allows users to place anchors representing groups of interest. The system then uses a spring simulation to generate a graph layout by animating group members' names gravitating towards the corresponding anchors. This results in a simpler spring simulation because springs only connect movable nodes to unmovable anchors. The VIBE system [7] uses a visualization for document retrieval with points of interest defined by key terms and a location on the display. The retrieved documents are placed based on their relevance to the points of interest using a force model.

Dontcheva et al. [3] use a spring layout algorithm to indicate overlap in tags among photos returned via queries. Two photos have a spring between them if they share a tag. The layout is computed once in response to the photos returned by the query. Users are limited to highlighting and pruning

IUI'09, February 8–11, 2009, Sanibel Island, Florida, USA.

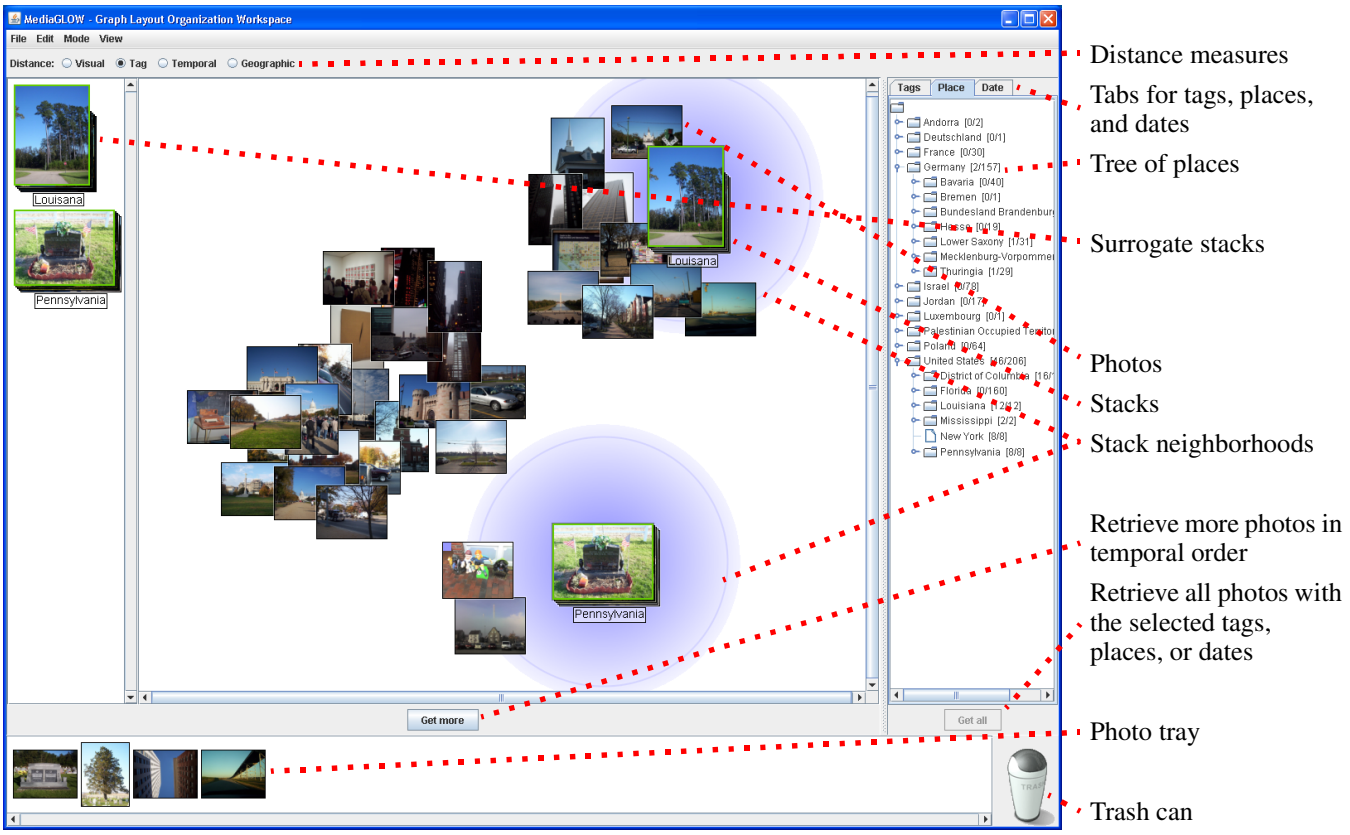

Figure 1. MediaGLOW after creating two stacks.

photos. This removes photos and springs connected to them but does not adjust the lengths of springs. Users cannot pin or drag items around and are not able to create stacks or groups that are represented as a single item in the network.

MEDIAGLOW

MediaGLOW is an interactive, two-dimensional, graph-based workspace that supports users in locating and organizing photos (see Figure 1). Unlike grid- and tree-based presentations, the workspace places related photos near each other without being constrained to a grid. Instead of only allowing users to browse through time-ordered photos linearly or through photos placed on a map, our system enables alternate modes of exploring the collection. By grouping photos in stacks based on multiple similarity measures and direct feedback, users can locate photos related via multiple features, enabling flexible, adaptive browsing.

Graph-based Workspace Layout

To help users organize photos into meaningful groups, the workspace maintains a graph representing the similarity distances between photos and stacks of photos. Each graph node is either an individual photo or a stack of photos. The graph is fully connected in the absence of stacks. Stacks have neighborhoods of similar photos that form independent fully connected graphs with no outgoing connections.

There are several advantages to using a graph layout model. First, because only distances are needed, a space with non-Euclidian distances can be used. Second, unlike self-organizing maps [5], nodes can be placed in any position, not just in a grid or similar structure of cells. Third, a graph layout can take placements by the user into consideration. This differentiates it from approaches such as multi-dimensional scaling [1]. In addition, by using a distance metric with partial derivatives that guides the direction in which the iteration moves, a satisficing state can be found quickly.

We use a spring model to determine a layout in which the spring system is in a state of minimal energy. Such a layout places related nodes, i.e., nodes with high similarity, near each other and unrelated nodes far apart from each other. Figure 1 shows a layout of individual photos and two stacks.

To produce the initial layout, photos are placed in random positions and then iteratively moved in small increments until a state of low energy is reached. The spring model is started in multiple random states and refines the state with the lowest energy to avoid finding a single local minimum.

The layout time is proportional to the square of the number of graph nodes (about 3µs per square-node on a 3GHz Intel Core 2; 0.12s for 200 nodes; 3s for 1000 nodes), where a photo stack counts as a single node. Layout time is further reduced because a stack has severed connections with graph nodes outside its neighborhood. Updating the layout during dragging is feasible for up to 200 nodes and updating at the end of dragging feasible for up to 1000 nodes.

Distance Measures

For determining the desired distances between nodes, different distance measures can be used. A simple but useful distance measure is the difference between the photo creation times. If photos have location data, the geographic distance

Figure 2. Tool tip and similar photos.

between the locations can be used as a distance measure. Different distance measures produce different patterns in the workspace. For example, when organizing the workspace by a temporal distance measure, photos are grouped in strings of tight clusters whereas a visual distance measure tends to more evenly distribute photos.

We have used a variety of distance measures including temporal, geographic, and visual distances. Our visual similarity measure takes advantage of manually tagged data available at the Flickr photo sharing site. Classifiers based on visual features are trained on tagged photos to predict the likelihood for a set of tags. This vector of tag probabilities for two photos is used to compute the Kullback-Leibler distance [6]. Another distance measure is based on the user-assigned tags that is computed as the Jaccard distance of tags shared across photos.

Spring Model

Springs between all pairs of graph nodes determine the layout of the photos. The neutral lengths of the springs are determined by the chosen measure of similarity between photos. A distance threshold determines if photos would be in the neighborhood of a stack. On mouse-over of a graph node, all spring connections to that node within that distance threshold are drawn as blue lines. The nodes connected via those connections are highlighted. Figure 2 shows a close-up of a graph with the mouse hovering over one photo. As can be seen, most of the similar photos are nearby with less similar ones placed further away.

There are different approaches for creating a layout for the completely connected graph to optimize the distances between nodes with respect to the desired distances. In our system, we chose to use a force model for the graph layout. Such an approach optimizes the linear error of distances. The force is modeled by springs using Hooke's law. Specifically, a spring is defined by a neutral length l and a constant k. Two nodes at distance d connected by a spring (l, k) are subject to the force $k * (d - l)$. This is a repulsive force if $d <$ l. The length of the springs at rest corresponds to the desired distance between the nodes. Our spring model assumes a fully connected spring system (i.e., a spring between every pair of nodes), although the workspace is partitioned into several fully connected subgraphs. Users can pin photos to a position to prevent the system from moving them.

Normalizing Spring Lengths

Many distance measures do not lend themselves to be used directly as spring lengths. Both temporal and geographic distances have large gaps where no photos were taken. On the flip side, visual distances tend to cluster in the middle of the range. To deal with this issue, we uniformly distribute spring lengths across the whole range. Normalization of distances is performed by sorting all pairwise distances and by determining the percentile for each distance value. A desired distance at the 10 percentile of all distances would be assigned a spring length of 0.1 and a desired distance at the 80 percentile would be assigned a spring length of 0.8.

For a better separation between nodes, we specify a minimum spring length l_{min}. Spring lengths are modified using this formula: $l' = l_{min} + l * (l_{max} - l_{min}) / l_{max}$ where l_{max} is the maximum possible spring length between two nodes, usually one in a normalized space. We found that just specifying a minimum spring length provided insufficient separation and added an extra step after the spring layout that pushes apart nodes closer than the minimum spring length.

Zooming in the Workspace

Users can zoom in or out of the workspace. If photo positions and thumbnail sizes were adjusted by the same zoom factor, users would not be able to separate overlapping photos. On the other hand, if the zoom factor only applied to photo positions and the image sizes remained constant, users could not get a better look at the thumbnails. To address this issue, the system includes two different zoom factors for photo positions and thumbnails. Specifically, the system uses the square root of the workspace zoom factor as the zoom factor for the thumbnails for photos and stacks. Thus, if distances between photos are increased by a factor of 4, thumbnail sizes are only increased by a factor of 2.

Photo Stacks and Neighborhoods

To organize related photos, users may group them into stacks that are pinned in place and represented by a single keyframe or collage of keyframes. Stacks are surrounded by neighborhoods of similar photos. The extent of a neighborhood is visualized with a halo. The primary means that users have to organize the workspace are creating stacks and adding photos to stacks. Figure 3 shows a workspace with three stacks and their neighborhood halos.

Photo Stacks

Creating a stack results in a new element in the graph layout that replaces the component photo(s) and determines desired distances to other photos and stacks. As more photos are added to a stack, the system adapts those distances. Our implementation uses the average pairwise distance between stack members and other graph nodes.

In a pilot study, people found it difficult to locate stacks when zoomed in on a part of the workspace. To address this,

Figure 3. Neighborhood Halos

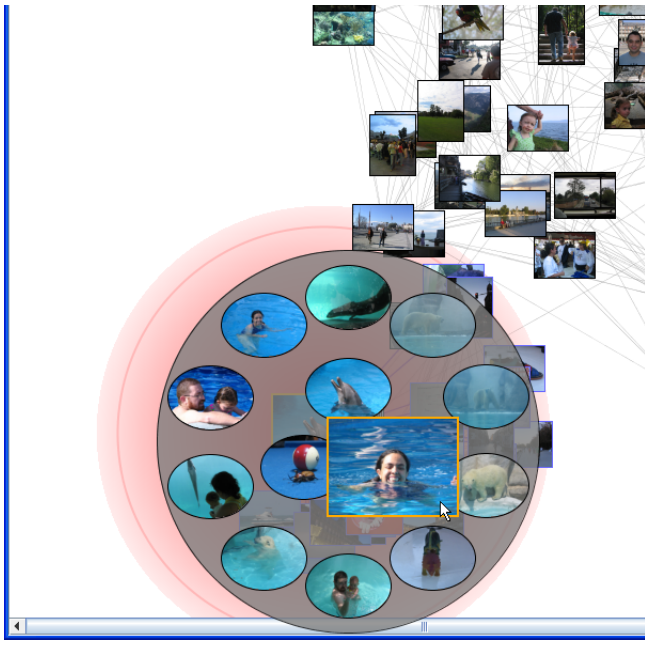

Figure 4. Expanding a Stack.

stack surrogates are also shown in a side bar to the left of the workspace (see Figure 1). All operations for stacks in the workspace can also be performed with their surrogates. The side bar can be hidden to see more of the workspace.

Users can explore the members of a stack by expanding it. This displays all thumbnails in the stack in a circular arrangement in one or more rings (see Figure 4). To make thumbnails fit better into the circular arrangement and for aesthetic reasons, we display just the part of the photo inside the inner bounding ellipse. When the user moves the mouse over one of these thumbnails, it is enlarged and the complete image is shown. Users may remove photos from a stack by dragging thumbnails outside the circle of the expanded view. Photo stacks and individual photos may be viewed with a conventional viewer popup in the workspace that allows the user to flip through all elements in a stack.

Stack Neighborhoods

As stacks are created, the system identifies photos that are close to a stack and indicates their potential inclusion in that grouping by placing them in the stack's neighborhood, or region of influence. We define a threshold d_n that describes the maximum distance between a photo and a stack such that the photo may belong to the neighborhood of the stack. A stack neighborhood is visualized as a circular halo with a radius equal to the threshold d_n. This provides a visual indication as to which nodes are in the neighborhood of a stack.

Photos may be highly similar to more than one stack. If those stacks are far apart, the photos would pile up along the straight line between the two stacks. To address this, we allow nodes to belong to only one stack neighborhood at a time. Photos that are close to more than one stack are placed in the neighborhood of the stack to which they are the closest based on the distance metrics of the stacks.

For each neighborhood, all spring connections leading out of the neighborhood are cut. This allows a stack and its neighborhood of photos to be dragged to a new location without altering the layout of other stacks or photos.

Neighborhood Halos

Stack halos are generally all the same color (blue in the current implementation), reflecting the fact that photos can only belong to one neighborhood regardless of their similarity to other stacks. When the user moves the mouse over a stack, the system visualizes overlap in potential neighborhood membership by color-coding other stack neighborhoods by the degree of potential overlap with the current stack neighborhood (see Figure 3). Strongly related neighborhoods are shown with a red halo (hot) and very unrelated neighborhoods are shown with a blue halo (cold).

For determining how similar two neighborhoods are, we look at the number of photos that could belong to both neighborhoods. Their similarity is measured with the Jaccard coefficient, i.e., the size of the intersection of neighborhoods divided by the size of the union of neighborhoods. The halo visualization indicates which stacks are closely related from the system's view and thus are likely locations where the user should look for elements that have been assigned to the wrong neighborhood by the system.

Adaptive Distance Measures for Stacks

The iterative process of establishing user-defined categories by grouping related photos into stacks benefits from the adaptation of the workspace. The spring layout with neighborhoods and the visualization of potential neighborhood overlaps are simple forms of adaptation to user actions in the workspace. More complex forms of adaptation are employed to develop specialized distance metrics for each stack and to redefine the overall distance measure based on the stacks created.

In addition to the distance metrics described above, we define a distance metric for a stack based on the features

most representative of its elements. To identify these features, the system determines an approximation of the shortest path through all values of a feature and computes the mean and standard deviation of the distances between adjacent values on this path. Adjacent distances are used instead of all pairwise distances to avoid the domination of large gaps that would be included in many pairwise distances.

The system computes the mean of the adjacent distances for the stack members. Features where the local mean is either more than one standard deviation below the overall mean or less than 25% of the overall mean have their weights increased in the calculation of desired distances to the stack. For example, if the mean adjacent temporal distance is 11 days with a standard deviation of 26 days, all stacks where the mean adjacent distance is below 2.75 days (25%) get an increased weight for the temporal feature.

Because each stack ends up with its own distance metric, the workspace can effectively represent categories that use widely varying subsets of the feature space for their definition. Thus, time, location, visual similarity, and tags can be included in some distance calculations and not in others. This approach generalizes well as additional distance measures such as face detection are added to the system.

Selecting and Retrieving Photos

Photos can be selected in the workspace with standard selection gestures as well as by using lists and trees of tags, places, and dates (see the right side of Figure 1). In the trees of places and dates, both leaves and intermediate nodes can be used for selections. Multiple selections are also possible, for example, selecting two countries to select all photos taken in either country.

Users may retrieve additional photos that are similar to a selected set of photos or stacks using visual, temporal, or geographic distance measures. Photos can also be retrieved using the "Get all" button for a particular tag, place, or date. To avoid large changes in the workspace layout, newly added photos are placed in random locations in the workspace and then iteratively moved to locations with lower energy. After they have settled into satisfactory positions, both old and new photos are moved iteratively to find an overall low energy state. This two-stage approach keeps old photos near their previous locations to minimize user disorientation.

USER STUDY

A user study was performed to explore users' reactions to different features of MediaGLOW and to gain insight for design modifications. To provide a benchmark, we also asked them to use a more traditional photo organizer that groups photos in a grid-based light table by date, place, or assigned tags [4]. We gave participants a set of 450 geotagged photos from a variety of locations over the course of a year. They were asked to select and place photos into five categories: nature, cuisine, nightlife, culture, and architecture. They then chose three photos from each of the five categories to determine 15 photos to be used in a travelogue.

The overall results show that the traditional scrolling light table interface was more efficient for the task of categorizing and selecting photos than the dynamic layout approach of MediaGLOW. One bright point for MediaGLOW was that it was considered more fun by seven of the participants, with one being neutral. Comments from participants indicated that many thought the workspace, stacks and expanding stacks were good ideas in general. The big negative was the automatic spring layout making it difficult to keep track of progress and cluttering the workspace.

CONCLUSIONS

The innovation with MediaGLOW lies in an interactive workspace that combines visual metaphors and user interface techniques with image classification and 2D similarity-based layout techniques. MediaGLOW seeds the user activities by grouping similar photos. Users may choose among visual, tag, temporal, and geographic attributes to locate similar photos. A graph-based layout algorithm, where photos are graph nodes, automatically places similar photos near each other in the workspace. Users locate, tag, refine, differentiate, and collect photos using a number of workspace visual cues: photo stacks, stack keyframes, and neighborhood halos. Our system goes beyond traditional browsing interfaces offered by photo organization systems and allows users to focus on sets of photos for which they can locate related photos along several dimensions.

The system presented here has been positively received by early users. They described it as a fun, aesthetically pleasing, and interactive learning workspace. With a good distance measure, the system produces intuitive groupings of photos. Users are engaged when interacting with the system. They are able to organize photos in meaningful ways. The study also suggested improvements for MediaGLOW.

REFERENCES

1. Borg, I., Groenen, P. (2005). *Modern Multidimensional Scaling: theory and applications* (2nd ed.), Springer-Verlag New York.
2. Donath, J.S. (1995). Visual Who: Animating the Affinities and Activities of an Electronic Community. *Proc. of ACM Multimedia*, New York: ACM Press, 99-107.
3. Dontcheva M., Agrawala M., Cohen M. (2005). Metadata Visualization for Image Browsing. *UIST 2005 Demonstration*.
4. Girgensohn, A., Adcock, J., Cooper, M., Foote, J., Wilcox, L. (2003). Simplifying the Management of Large Photo Collections. Human-Computer Interaction INTERACT '03, IOS Press, pp. 196-203.
5. Kohonen, T. (2001). *Self-Organizing Maps*. Berlin: Springer Verlag.
6. Kullback, S. (1987). The Kullback-Leibler distance. *The American Statistician* 41, 340-341.
7. Olsen, K.A., Korfhage, R.R., Sochats, K.M., Spring, M.B., Williams, J.G. (1993). Visualization of a Document Collection: the VIBE System, *Information Processing & Management*, 29(1): 69-82.
8. Platt, J.C., Czerwinski, M., Field, B.A. (2003). PhotoTOC: automatic clustering for browsing personal photographs. *Information, Communications and Signal Processing*, pp. 6-10.