Mobile Photo Album Management with Multiscale Timeline*

Kolbeinn Karlsson†
Huawei Media Lab
Bridgewater, New Jersey
kk752@cornell.edu

Wei Jiang
Huawei Media Lab
Bridgewater, New Jersey
wei.jiang@huawei.com

Dong-Qing Zhang
Huawei Media Lab
Bridgewater, New Jersey
dongqing.zhang@huawei.com

ABSTRACT

Traditional photo browsing systems developed for PCs are inefficient for browsing and searching of large photo albums on mobile devices due to the small screen size and limited mobile processing power. We propose a new concept in this paper, the multiscale timeline, where photos are grouped into clusters and displayed sequentially on a scaled timeline with user controllable time scales, enabling multiscale overview of the photo album for efficient browsing and searching. To address the slow speed of re-clustering when new photos are added, a new incremental spectral clustering algorithm is further developed, which is an order of magnitude faster than the traditional spectral clustering algorithm and its conventional incremental version. Our implementation of the system on mobile devices shows a better user experience and browsing efficiency based on the experiments over large real-world photo collections.

Categories and Subject Descriptors

H3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems

General Terms

Algorithms, Experimentation

Keywords

Multiscale timeline; mobile album; incremental clustering

1. INTRODUCTION

With the prevalence of high-quality mobile cameras with large storage, we have witnessed an explosive amount of mobile photos. It is more and more popular that a large number of such photos are stored on the mobile device, and effective on-device photo management has become increasingly important. Traditional mobile photo browsers, *e.g.*, on iOS and Android systems, follow the file browsers in PCs, where photos in a folder are presented in a flat grid-based view, as shown in Fig. 1 (a). Such browsers can be very inefficient in dealing with large-scale albums. First, mobile devices normally have small screens, and it can be very hard for users to browse through the album or to locate individual photos. Moreover, users often take multiple near-duplicate photos of the same scene, *e.g.*, in burst mode. A flat display of all these photos on the small screen further reduces the efficiency of on-device photo browsing. Second, it is important to convey the photos' time order in a browsing system, but simply laying out a large number of photos in a flat timeline (as shown in Fig. 1 (b)) may not be efficient. It is even harder than in the grid-based view to swipe through the large album to locate photos of interest. Finally, in addition to the capture time, the photos' visual content is also useful for effective browsing and search. For instance, in cases where a lot of photos are captured close in time but with highly different content, *e.g.*, during a vacation, the visual content can help organize such photos more efficiently.

Large-scale photo album management has been extensively studied on conventional PCs, where large screens and strong computational powers are available. For example, systems like Time Quilt [4] and Photoland [12] organize photos based on visual features and metadata, and systems like PhotoMesa [2] and PhotoFinder [5] focus on screen layout and/or user queries. Commercial solutions like Apple's *iPhoto* and Google's *Picasa* rely heavily on user tags and manual sorting. None of these systems, however, can easily be used for on-device mobile photo management, due to the limited screen size and processing power on mobile systems.

To facilitate photo browsing on mobile devices, some on-device presentation systems have been developed to arrange photos on non-uniform grids [11] or on three-dimensional structures [1, 15]. However, the underlying photo organization remains unchanged. Some other work aims at both photo organization and presentation [6, 7, 8] by clustering photos based on time, faces, and background features. However, the clustering result is still presented in an inflexible grid-based view or a simple flat timeline.

Addressing the characteristics of on-device mobile photo browsing, we propose a ***multiscale timeline*** framework to effectively organize and browse photos based on both time and visual content. Fig. 1 (c) illustrates the concept of the multiscale timeline. Each time scale corresponds to a certain

*Figures 1 and 3 modified from original submission due to copyright restrictions.

†Currently with Cornell University.

MM'14, November 3–7, 2014, Orlando, Florida, USA.
Copyright 2014 ACM 978-1-4503-3063-3/14/11 ...$15.00.
http://dx.doi.org/10.1145/2647868.2655060.

(a) grid-based view

(b) single flat timeline

(c) multiscale timeline

Figure 1: Comparison of different browsing schemes.

level of granularity to partition the album, where photos are clustered at that level for users to browse. The multiscale timeline provides multiple granularities to organize photos, where users can quickly switch among different time scales to examine different levels of details of the album. For efficient browsing, each cluster is presented as a thumbnail consisting of a set of automatically selected representative images from the cluster. To accommodate the limited processing power in mobile devices, a multiscale one-step incremental spectral clustering algorithm is also proposed to quickly cluster photos at multiple time scales with a complexity approaching $O(n)$. To further improve browsing efficiency, near-duplicate photos are automatically detected based on both time and visual features, and are collapsed to a single representation to save space on small screens. We implement our system on an Android tablet. Evaluation over two large consumer albums from Flickr demonstrates the effectiveness of our multiscale timeline system.

2. MULTISCALE TIMELINE ALBUMING

Many previous photo clustering algorithms attempt to cluster photos by events in order to categorize the photos in a way that the user would do. A major problem, however, is that events are inherently multiscale. For instance, an event could be a group of people posing in front of a camera, a day spent at a theme park, or a week-long vacation. It is not only difficult to predict which scale the user would want to use to sort the album, but the scale might also change depending on what the user is looking for at the moment. Therefore we cluster the photo album on multiple scales and allow the user to browse all of them. To efficiently browse the clustering results we introduce the concept of multiscale timeline clusters, which is illustrated in Fig. 2. On each scale, the clusters (*i.e.* their representative thumbnails) are placed sequentially on a timeline that the user can browse using the pan and flick gestures. The user can also quickly change the time scale using the pinch gestures, as though to "zoom" between scales. The user can inspect the contents of a cluster by tapping its thumbnail. The contents of the inspected cluster is presented as either a traditional grid-based view or as another multiscale timeline depending on whether or not the photo number in the cluster exceeds a predetermined threshold. The multiscale time line clusters are created by extracting time and content-based features from the photos and performing an efficient one-step incremental clustering algorithm on the extracted feature sets.

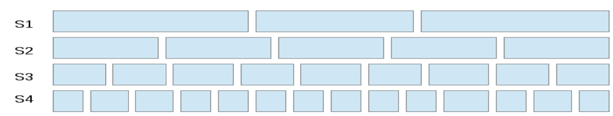


Figure 2: Multiscale clustering of photos (scales S1-S4), which supports multiscale timeline browsing.

2.1 Features and Near-Duplicate Detection

Both time and visual features are used for clustering and browsing the photo album, which effectively complement each other to improve the system performance. The timestamp of each photo is extracted from the metadata, and three types of visual features are extracted from each image: A bag-of-words representation using SURF local descriptors (SURF BoW), a 64-bin uniform *Lab* color histogram, and the global grid-based color moment (GBCM) also in *Lab* color space. The SURF BoW uses a codebook of size 500, built by K-means clustering of randomly sampled SURF descriptors over the album. The 225-dimensional GBCM feature consists of the first three color moments computed over 5x5 image grids.

For fast near-duplicate detection, we use a cascade of binary classifiers similar to [13], based on both time and visual features. First, by comparing timestamps of image pairs, we quickly rule out most pairs that could not possibly be near-duplicates. Then a relatively fast binary classifier using GBCM is used to further rule out non-near-duplicates. The remaining pairs are finally classified by a binary classifier using the SURF BoW features. All these classifiers are trained offline using a set of ground-truth near-duplicate images and are tuned towards a low rate of false positives, so that the chance of collapsing non-near-duplicate images is kept very low. Each group of near-duplicate images is treated as one image for clustering and browsing, whose features are the average timestamp and visual features of the entire group.

2.2 Incremental Multiscale Clustering

Assuming that we have N different scales, the initial multiscale clustering is conducted over each scale individually based on the spectral clustering algorithm [9]. After that, when a new image is added, a fast one-step incremental spectral clustering algorithm is performed.

2.2.1 *Overall spectral clustering*

On each scale, a sparse similarity matrix $\mathbf{S}$ of the entire photo album is constructed based on an aggregated similarity measure that is the convex combination of the K similarity functions defined on time and visual features:

$$S(y_i, y_j) = \sum_{k=1}^{K} \alpha_k S_k(y_i, y_j) \tag{1}$$

where $\alpha_k \geq 0$ for $k = 1, \ldots, K$ and $\sum_{k=1}^{K} \alpha_k = 1$. Each entry $S_k(y_i, y_j)$ in $\mathbf{S}$ is the similarity between point y_i and point y_j for each feature, based on the Gaussian kernel:

$$S_k(y_i, y_j) = \exp\left(-a_k d_k(y_i, y_j)^2 / \sigma_k^2\right) \tag{2}$$

where $d_k(\cdot, \cdot)$ is the appropriate distance metric of feature k. Both the constants a_k and the scaling factors α_k are varied across different scales in such a way that, for larger scales, the time difference dominates, since photos taken far away from each other normally do not share similar contents and should not be clustered together. As the scale becomes smaller, the weights of the content features increase. All similarity values below a sparsity threshold are set to zero.

Following the recipe of [9], the initial spectral clustering is performed by solving the generalized eigenvalue system:

$$Lx = \lambda Dx \tag{3}$$

where L is the graph Laplacian computed as $L = D - A$; A is a similarity matrix whose entries are $A_{ij} = S(y_i, y_j)$ if $i \neq j$ and $A_{ii} = 0$; and the degree matrix D is diagonal with $D_{ii} = \sum_j A_{ij}$. L and D are both symmetric. The number of clusters is automatically determined from the eigengap [14].

2.2.2 *One-step incremental clustering*

Even for a sparse L, efficient algorithms such as the Lanczos method can still require $O(n^{1.5})$ complexity to solve Equation (3). For mobile users, reclustering the entire album everytime a new image is added can be unbearable, especially for large n. To avoid frequent reclustering, we propose an incremental spectral clustering algorithm based on a well-known result from perturbation theory [10]. Let ΔL and ΔD denote the perturbations (change) to L and D, respectively. Then the first-order perturbation of the eigenvalue, $\Delta\lambda$, is given by:

$$\Delta\lambda = \frac{x^T(\Delta L - \lambda \Delta D)x}{x^T Dx} \tag{4}$$

It is shown in [10] that the perturbation of the eigenvector, Δx, can be obtained by solving the following linear system:

$$\phi \Delta x = h \tag{5}$$

where $\phi = L - \lambda D$ and $h = (\Delta\lambda D + \lambda \Delta D - \Delta L)x$. Let's assume that at a certain stage, n points have been clustered already. When a new data point y_{n+1} is added, the perturbation ΔL is the difference between the new Laplacian, L_{n+1}, and a zero-padded L_n:

$$\Delta L = L_{n+1} - \begin{bmatrix} L_n & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} B & -b \\ -b^T & b^T \mathbf{1} \end{bmatrix} \tag{6}$$

where $B = \text{diag}(b)$, $\mathbf{1}$ is a vector of all ones, and b is the n-dimensional vector whose i-th entry is the similarity between point y_{n+1} and y_i. Likewise, the perturbation of the degree matrix can be decomposed the same way:

$$\Delta D = D_{n+1} - \begin{bmatrix} D_n & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} B & 0 \\ 0 & b^T \mathbf{1} \end{bmatrix} \tag{7}$$

By putting Equations (6) and (7) into Equation (4), we get:

$$\Delta\lambda = (1 - \lambda) \frac{x^T Bx}{x^T Dx} \tag{8}$$

As both B and D are diagonal, Equation (8) can be efficiently computed in $O(n)$. The perturbation method is only a first-order approximation, so error will build up over time. Therefore, the full clustering algorithm will be performed again after M updates, where M is determined empirically.

We name the above proposed method the one-step incremental clustering algorithm since only one step is required to update each eigenvector. This is in comparison with the original incremental spectral clustering method in [10], where the effect of adding each image is decomposed into a series of similarity changes in $\mathbf{S}$, each change corresponding to solving a sparse linear system for each eigenvector. As a result, our one-step incremental clustering is dozens of times faster than the method of [10] on average, based on our implementation.

2.3 Selecting Representative Images

When the user browses the photo album, each cluster will be presented by a thumbnail consisting of one or more automatically selected representative images. The thumbnail is constructed differently depending on the size of the cluster. For cluster sizes below a predetermined threshold T, the image with the highest degree centrality is selected as the representative image as suggested in [3]. If the clusters are larger than T, four images are selected to create a collage thumbnail by first dividing the cluster into four subclusters using spectral clustering, then finding the image with the highest degree centrality in each of the subclusters.

3. EXPERIMENTS

The proposed multiscale timeline albuming system is implemented in a Samsung Galaxy Tablet, a screen capture shown in Fig. 3. Users can use the pinch gesture or buttons on the screen to zoom in/out the time scales, and tap the cluster thumbnails to inspect images in a particular cluster. In our implementation, we cluster the data on four different scales (*i.e.* $N = 4$). The top scale is over the whole collection. For smaller scales, the image collection is divided up into chunks spanning a single month, week, or day, and each chunk is then clustered separately. An added benefit of such division is the further computation speedup, since spectral clustering has superlinear complexity.

Due to the privacy concern of using private mobile photos, we decided to create the test data sets using the images under Creative Commons License from Flickr, where the characteristics of many albums are very similar to those of personal albums on mobile devices. The selection criteria included time span, which should be several years and evenly distributed; and content, which should not include a lot of post-processed images. At last, two Flickr users satisfying the above requirements were selected among 25 candidates. Dataset 1 has 2092 photos taken over a period of 40 months, and dataset 2 has 3442 photos taken over a period of 13 months.

The performance of the incremental clustering algorithm is evaluated by comparing the clustering quality and speed with the regular spectral clustering algorithm when a new image is added. The quality is evaluated using normalized mutual information (NMI) compared with batch spectral clustering. That is, we use batch spectral clustering of the entire dataset as the ground truth and compare the incrementally clustered dataset at different stages to the ground truth using NMI. Fig. 4 (a) shows the comparison results. The NMI measures how similar the incremental result is to the batch result of the entire dataset. As we see in the figure, the incremental clustering becomes more unlike the ground

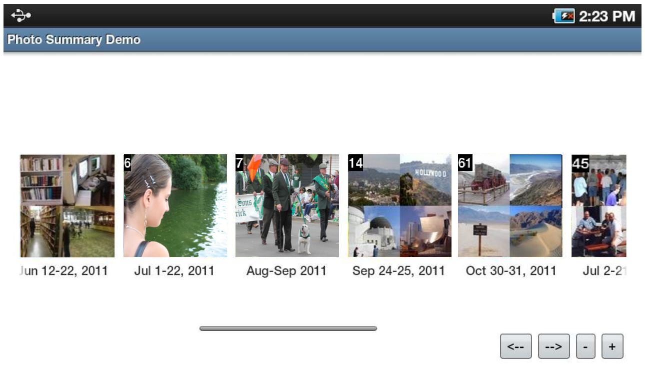
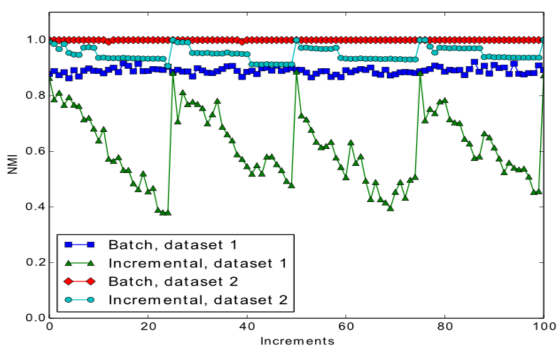
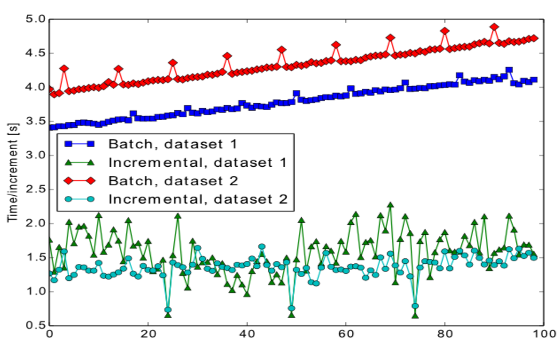


Figure 3: The screenshot of the implemented system

truth after each step due to the approximation of first order perturbation. This is remedied by rerunning batch clustering periodically to limit the error accumulation. The number of updates to run before rerunning the batch version is chosen to balance the trade-off between performance and speed. We see that the algorithm performs considerably better on dataset 2 than 1. The difference reflects the higher sparsity of dataset 1, since the system in eqn. 5 becomes more ill-conditioned as the Laplacian becomes sparser. Fig. 4 (b) shows the speed comparison of incremental clustering and batch clustering when one photo is added. The figure shows that while the time lapse per increment clearly increases with every increment for the batch version, the incremental version stays nearly constant.



(a) quality of clustering



(b) time per increment

Figure 4: Incremental spectral clustering versus batch clustering

4. CONCLUSION

A new album system is proposed for photo browsing on mobile devices, which typically have limited screen size and processing power. The multiscale timeline design is introduced to address the problem of limited screen size and for more efficient photo browsing. And a new incremental spectral clustering algorithm is developed to speed up the spectral clustering algorithm when new photos are added to the album. The implemented mobile application using the multiscale timeline concept results in a more efficient browsing experience, and the new incremental clustering algorithm significantly accelerates the speed of album reorganization when new photos are added on large photo albums. Future work may include refining the system for continuous time scale change, and further improving clustering accuracy.

5. REFERENCES

[1] D. Ahlström, and et al. A user study on image browsing on touchscreens. In *ACM Multimedia*, pages 925–928, 2012.
[2] B.B. Bederson. Photomesa: a zoomable image browser using quantum treemaps and bubblemaps. In *ACM UIST*, pages 71–80, 2001.
[3] W.T. Chu and C.H. Lin. Automatic summarization of travel photos using near-duplication detection and feature filtering. In *ACM Multimedia*, pages 1129–1130, 2009.
[4] D.F. Huynh, and et al. Time quilt: scaling up zoomable photo browsers for large, unstructured photo collections. In *ACM CHI EA*, pages 1937–1940, 2005.
[5] H. Kang and B. Shneiderman. Visualization methods for personal photo collections: Browsing and searching in the photofinder. In *IEEE ICME*, 2000.
[6] J. Kim, and et al. Photo cube: An automatic management and search for photos using mobile smartphones. In *IEEE DASC*, pages 1228–1234, 2011.
[7] K. Kim, S. Kim, and H.-G. Cho. A compact photo browser for smartphone imaging system with content-sensitive overlapping layout. In *ACM ICUIMC*, 2012.
[8] M. La Cascia, M. Morana, and S. Sorce. Mobile interface for content-based image management. In *IEEE CISIS*, pages 718–723, 2010.
[9] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering1 analysis and an algorithm. *NIPS*, 14:849–856, 2001.
[10] H. Ning, and et al. Incremental spectral clustering by efficiently updating the eigen-system. *Pattern Recognition*, 43(1):113–127, 2010.
[11] K. Ren and J. Calic. Freeeye: interactive intuitive interface for large-scale image browsing. In *ACM Multimedia*, pages 757–760, 2009.
[12] D. Ryu, and et al., Photoland: a new image layout system using spatio-temporal information in digital photos. In *ACM SAC*, pages 1884–1891, 2010.
[13] F. Tang and Y. Gao. Fast near duplicate detection for personal image collections. In *ACM Multimedia*, pages 701–704, 2009.
[14] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.
[15] S. Yousefi. 3d photo browsing for future mobile devices. In *ACM Multimedia*, pages 1401–1404, 2012.