Time as Essence for Photo Browsing Through Personal Digital Libraries

Adrian Graham, Hector Garcia-Molina, Andreas Paepcke, Terry Winograd
Stanford University
{adrian.graham | hector | paepcke | winograd}@cs.stanford.edu

ABSTRACT
We developed two photo browsers for collections with thousands of time-stamped digital images. Modern digital cameras record photo shoot times, and semantically related photos tend to occur in bursts. Our browsers exploit the timing information to structure the collections and to automatically generate meaningful summaries. The browsers differ in how users navigate and view the structured collections. We conducted user studies to compare the two browsers and a commercial image browser. Our results show that exploiting the time dimension and appropriately summarizing collections can lead to significant improvements. For example, for one task category, one of our browsers enabled a 33% improvement in speed of finding given images compared to the commercial browser. Similarly, users were able to complete 29% more tasks when using this same browser.

Keywords
Photo browser, image browser, Personal Digital library, time-based clustering, ACDSee, summarization, time-based navigation, burst identification

1. INTRODUCTION

Given the importance of multimedia digital libraries, several project have developed techniques for searching video, images, and geographic data (e.g. [1, 2, 3]). A majority of other digital library efforts focused on text, because text is most accessible to a variety of search and processing algorithms. Certainly, Personal Digital Library (PDL) facilities remained overwhelmingly textual. By PDLs we mean bodies of information that are mostly of importance to individuals or small groups.

Two changes over the recent years have made the development of PDLs with image content urgent. (i) Desktop machine storage has grown cheap and plentiful and (ii) entry-level, point-and-shoot digital cameras have reached large numbers of consumers who are not necessarily deeply computer-savvy. As digital cameras do not require expensive film that can only be used once, users tend to take large numbers of pictures. This in turn results in an image management quagmire similar to the shoebox phenomenon of paper prints. Professional photographers and news reporters face the same challenge.

There have been several approaches to improving the experience of browsing and managing collections of personal digital photographs. The most basic photo browser available is a file manager with integrated support for viewing images and thumbnails. Such functionality is found in Windows XP, Mac OS X, and standalone browsers such as ACDSee [4, 5, 6]. Although simple to use, these tools tend to be limited, and do little to help users organize their pictures more effectively than they can in the physical world.

Using a database to manage photographs is another approach to this problem, used in systems such as FotoFile and PhotoFinder [7, 8]. Although these systems offer powerful search functionality, they have failed to catch on, largely because they require time-intensive manual annotation.

PhotoMesa and iPhoto are both examples of photo browsers which automatically group images for display. PhotoMesa groups by folder, year, and month, and iPhoto groups by "roll" (batch of photos downloaded to a computer at once) [9, 10]. This approach helps give more structure to the images than an enhanced file manager, but requires none of the time-intensive annotation of the database systems.

Our approach to this problem builds on the idea of automatically grouping photographs. Instead of using metadata that is directly available in the file system, we analyze the photo creation process and construct a variety of organizational structures. For instance, we can automatically organize photographs by event, realizing that sequences of photos of one event will be taken closer together in time than sequences of other photos.

An additional design goal was to build a system that could scale to visually browse a lifetime's worth of photographs. Photo browsers are notoriously poor at scaling, often unable to provide useful visual information for more than a few hundred images. Through our summarization techniques, we are able to browse tens of thousands of photographs.

These techniques are part of the two-phased approach the Stanford Digital Library Project is taking to PDL image management. During the first phase, we are exploring algorithms that organize and summarize digitally created photographic image collections *without* any curator assistance, using current, mature technologies only. While heuristic and therefore imperfect, we are trying to push this area as far as possible. We are careful during this phase to

design our systems such that machine image analysis technologies can be integrated later on.

During the second phase we will examine how gracefully we can take advantage of incremental image captioning efforts that a curator might be willing to invest. The goal is for the collection's user interface to take advantage of coarse, as well as fine-grained captioning as curators create and refine such captions over time. In this paper we present the current state of our phase I work, which is implemented in prototypes.

We begin with a description of one of our two browsers, the *Calendar Browser*. Next, we describe the heuristic algorithms that underlie this browser. The subsequent section introduces our second *Hierarchical Browser*, and a commercial browser used for comparisons. Finally, we present our experimental comparisons.

2. CALENDAR BROWSER

The Calendar Browser (CB) takes advantage of photo time stamps. When taking a picture with a digital camera, the equipment records the date and time when the photo was taken, as well as technical details, such as aperture settings, distance from the focal plane, and whether a flash was used. This data is encoded into the image files, and is available to applications that access these files. The Calendar Browser (Figures 1a and 1b) is an intentionally extremely simple interface that uses these date/time stamps to enable drill-down browsing and summarization.

The application window is partitioned into two panes, the control panel and the display panel. The display panel contains space to hold a constant number of fixed-size thumbnails (in our implementation we always show up to 25 images). At any given moment, the user is viewing images from one single time granularity. For example, Figure 1a shows a screenshot of the Calendar Browser at a 'one-year' granularity. At this granularity, all images in the display panel will have been taken during the same year, in this case 2001. Usually, the collection will contain many more than the 25 images. The challenge, therefore, is to select representative images from among the 2001 subset of the collection. The images in the display panel are sorted by time and are labeled with time designations of the next-lower time granularity. In Figure 1a the labels are therefore the months of the year 2001.

Notice that not every month is represented with the same number of images. While the browser dedicates only one of the 25 precious display slots to each of January and February, four slots are invested in the month of August. We will explain the allocation strategy in the next section, but roughly speaking, the allocated space is proportional to the number of photos taken during the respective time interval.

The two buttons in the control panel allow users to move through time at the current granularity. In Figure 1a this means moving backward and forward by year.

The user 'drills down' into the collection by left clicking on an image. (Right clicking moves us in the reverse direction.) Figure 1b shows the screen after the user left clicked on a June image. Notice that the control panel now indicates that the time granularity has changed to the month level. The control button labels have correspondingly been changed to 'previous month' and 'next month'. Images in the display panel are now labeled by days within the month of June when the respective photos were taken. Again, not every day is represented by the same number of images, but all of the 25 slots are filled.

Inspecting the display panel of Figure 1b further, we find a clue to how representative images are selected for a 'next-coarser' time granularity level. Notice that the largest sequences of image slots in Figure 1b are allocated for the 17th and the 30th of June, which contain five and four

Figure 1a: Calendar Browser, Year View

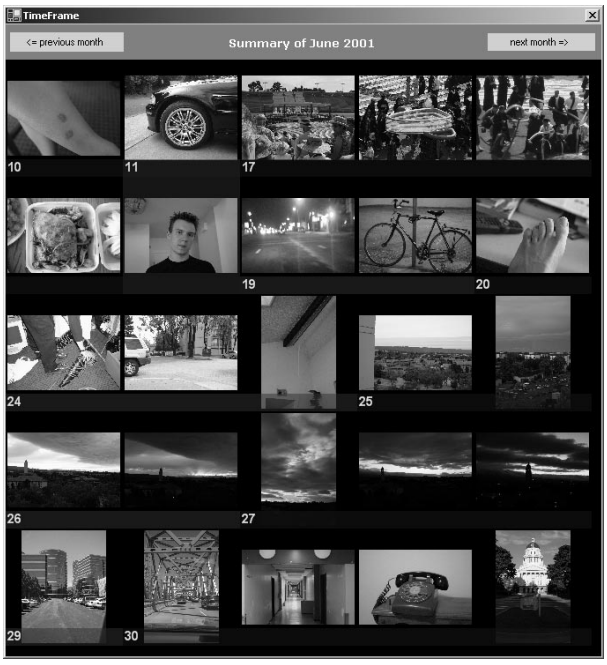

Figure 1b: Calendar Browser, Month View

images, respectively. The 24th and the 27th are tied with three images. Referring back to Figure 1a, notice that one image from each of the two largest June sequences, and one image from the 27th were picked to represent June for the year 2001.

Users may now left click on an image in one of the June days to see a summary of the photos that were taken on that day. If a user left-clicks on an image at the day granularity, that image will be placed in the center of the display panel, and the space around the image will be filled with the photos taken immediately before and after the center image. No summarization is applied at this level. Users may navigate using the previous and next buttons, to move through the un-summarized photographs 25 images at a time.

Earlier versions of this browser had several more controls that afforded the sliding of time windows and other powerful manipulations. Preliminary, informal user tests made it very clear that we were best off with the exceedingly simple design exemplified in the figures. Both novices and experts preferred this simpler interface.

3. TIME BASED SUMMARIZATION

People tend to take personal photographs in bursts. For instance, lots of pictures may be taken at a birthday party, but few, if any, pictures may be taken until another significant event takes place. We can derive information about personal collections of photographs based on these irregular, “bursty” patterns, which we analyze through clustering techniques.

The goal of clustering is to expose the structure that is present in a set of personal photographs as a result of the way the user has taken the pictures. Without realizing it, the user gives structure to his image collection by the pattern in which he chooses to take the photographs. Based on these patterns, we can organize the images without any further user intervention. For instance, someone may take a series of photographs at a birthday party. By identifying this burst, we can group these images together, thus creating a structural unit. We call this structural unit a cluster.

In many cases, we can derive even more specific information about the set of photographs in a cluster. Much as we were able to identify the "Birthday burst" by noticing an increased rate of activity, we are able to identify "sub-bursts" by comparing relative rates of activity within the cluster. For example, the rate at which pictures are taken is likely to be higher when someone is opening presents or blowing out candles on a cake. By encoding these sub-events within our cluster structure, we gain an even more accurate portrayal of the collection.

More generally, we can say that the burst structure within collections of personal photographs tends to be recursive, where bursts make up bursts etc. We represent this recursive burst structure using a tree of clusters, where photographs are stored only at the leaf nodes.

Figure 2 shows an example excerpt from a cluster tree. The labels of the nodes are added here for clarity. Our system does not generate them automatically. Each cluster node's internal 'label' is the time span that the images underneath the node cover.

Our clustering machinery is the substrate that underlies both phases of our project. First, we can query the cluster structure for information about the patterns present in the collection, such as: the number of images in a cluster, the number of clusters in a time span, the time span of the images within a cluster etc. Based on this information, we can create summarization schemes for a whole class of browsing interfaces. For instance, the Calendar Browser introduced above queries the cluster structure when it populates the screen at different time granularities.

Second, instead of operating on an image at a time, we are able to create user interfaces that operate on entire clusters. This will allow us, for example, to build an interface for incrementally adding metadata to a collection (Phase II of project).

Figure 3 shows the architecture that underlies the Calendar Browser. The raw image files at the bottom are accessed by the Metadata Extraction module. In our case, we extract the time stamps from the digital camera files. As global positioning systems (GPS) are integrated into digital cameras, the location where a photograph was taken could

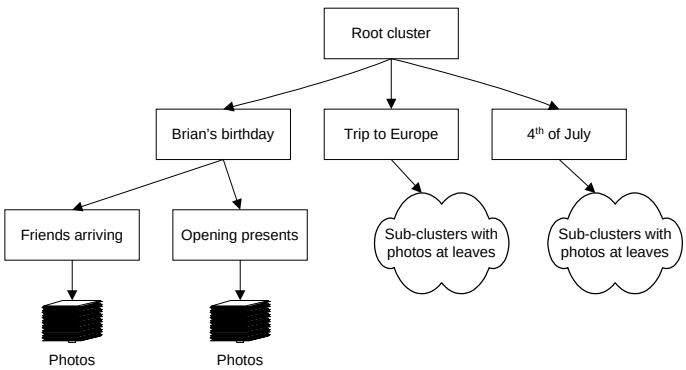

Figure 2: Example of Photo Cluster Tree

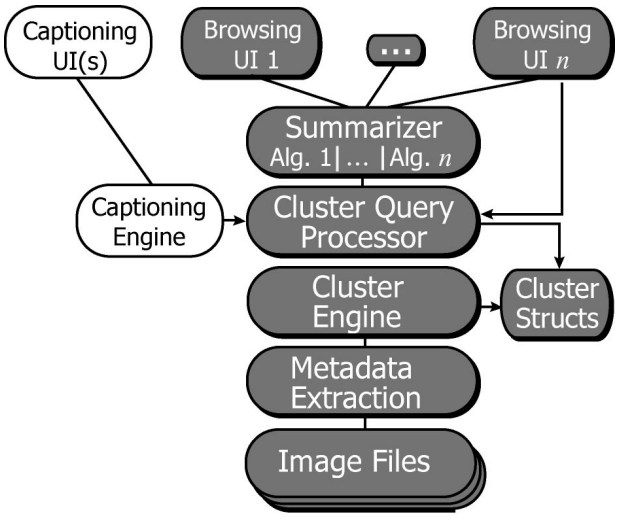

Figure 3: Browser system Architecture

also be extracted from the image files and introduced as a clustering criterion.

Alternatively, one might acquire an image analysis tool that extracts useful metadata other than what is stored textually within the image files. For example, if we integrated a tool that detects faces in images, we could enable clustering of images by whether they contain one or more people.

The Cluster Engine uses the extracted metadata to compute clusters. Our time-based engine, for example, computes the cluster tree introduced earlier. It is equally easy to cluster the data based on location when GPS data is available. The engine's output is a store of Cluster Structures, such as the cluster tree.

Client applications access the Cluster Structures store through a Cluster Query Processor. This processor solves queries such as 'find all images taken during July 4, 2001.' Browsers may submit queries to the Cluster Query Processor directly.

Another of the Cluster Query Processor's clients is the Summarizer module. In our case, the module uses time to compute the set of images that are to populate the browser's display panel. Other summarizer algorithms might base screen allocation decisions on other criteria.

The top layer of the architecture contains the photo browsers, of which the Calendar Browser is one example. The currently unimplemented captioning machinery will be another client (shown in white).

3.1. Time Cluster Engine

There are many ways to cluster images based on time metadata. For instance, one might use a K-means algorithm to cluster photos taken at about the same time. Another approach is to use time data to help improve the accuracy of clustering photographs using image analysis. AutoAlbum is a clustering method that works this way [11].

Our goal is to cluster photographs according to the burst patterns discussed earlier. The goal is achieved by grouping pictures that are taken at about the same rate. K-means clustering is difficult to use in this case because the specific number of bursts is not known. As well, the clusters AutoAlbum creates are not representative of bursts, since bursts are likely to contain pictures that are visually diverse. However, the basic time clustering present in AutoAlbum serves as a useful starting point for clustering by burst.

In order to determine where a burst begins and ends, we need to know the rate at which pictures were taken within the burst. However, in order to find out this rate, we need to know when the burst begins and ends. We overcome this cyclical dependency by initially clustering the images by a constant time difference. That is, we compare consecutive photographs, and if they differ in time by more than a specified amount, we create a new cluster. This gives a reasonable approximation of medium-sized clusters. We then create new clusters by splitting and combining these initial clusters. We next describe this three-step process in more detail.

Step 1: Create initial clusters.

We sort the list of photographs in the collection with respect to time, and create the root node of the cluster tree, which is empty. We create the first child cluster and add it to the tree. We then iterate through the sorted list of photographs. Every time two consecutive photographs differ by more than a specified constant time difference, we create a new cluster and add it to the root cluster. Regardless of whether we just created a new cluster, we add the current image to the most recently created cluster. By the time we reach the end of the list, we have a root cluster with as many child nodes as there are initial clusters. This simple, one-tiered cluster tree represents an approximation of the medium-sized bursts in the set of photographs. (This process is equivalent to the time-based clustering used in AutoAlbum [11].)

Choosing a good initial time difference depends on the collection of photographs. In general, more densely populated collections require smaller initial time differences (1 to 4 hours) for optimal clustering. Less dense collections are less dependent on this initial clustering step, and may in fact benefit from slightly larger initial time differences (8 to 24 hours). It should be possible for experienced users to set this parameter. However, in our current prototype it is fixed at four hours, which usually works well.

Step 2: Split initial clusters.

In Step 2 we want to further refine these clusters, by splitting them into appropriate sub-clusters (i.e. adding child nodes to the initial clusters).

During clustering, Step 1 could not take into account the rate at which photographs were taken within a cluster. Now that we have approximate clusters, we are able to consider these intra-cluster rates. This enables more accurate clustering, as the rate at which photographs are taken is likely to differ significantly between types of events. For instance, while taking a hike through a forest, someone may take a picture every couple of minutes. In contrast, when photographing a newborn baby for the first time, the time between pictures is likely to be in seconds.

We take these differing rates into account by comparing each pair of consecutive photographs to the ‘basic’ (see below for details) photographic rate of the cluster. When we come across a pair with a time difference that appears to be outside the normal range for the cluster (an outlier), we create a new cluster. This new cluster contains the images between the previous outlier (or the beginning of the cluster if no previous outlier exists) and the new outlier. This new cluster is then added to the original cluster as a child node. Since photographs are only stored in leaf nodes, the photographs contained by the new cluster are removed from the original cluster.

Figure 4 shows a snapshot of this stage in the process. Initial cluster 1 has been subdivided into new clusters 1.1

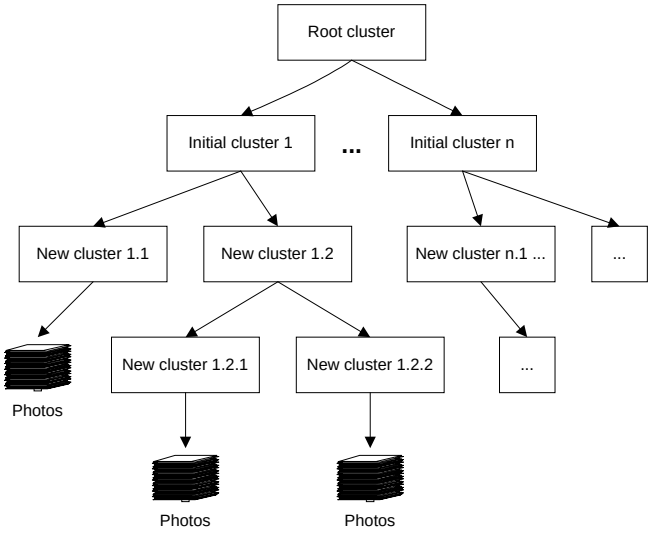

Figure 4: Partially Completed Cluster Tree

and 1.2. Cluster 1.2 still contained image bursts that were separated by significant spans of time, so it has been subdivided again. In contrast, cluster 1.1 now contains only images taken during a single burst of picture-taking.

Since bursts in collections of personal photographs tend to be recursive, we repeat this process over each newly created cluster until no outliers remain.

To find outliers in a cluster, we compute all the time differences between consecutive photographs. We then find Q_1, the value that divides the lower 1/4 of the values from the upper 3/4, and Q_3 that divides the lower 3/4 from the upper 1/4. A difference is an outlier if is greater than $Q_3 + 2.5\times(Q_3-Q_1)$. (The value 2.5 was selected empirically.) For each outlier, we split the cluster at the time that the large difference occurred.

Step 3: Create parent clusters.

This step is much like Step 2, except that instead of splitting initial clusters into more specific clusters, we combine initial clusters into more general clusters. One way is to use a fixed year-month-day hierarchy for the higher levels. That is, for each cluster C discovered in Step 1, we determine the date of its first photograph, and then we make C a child of the appropriate day in the year-month-day hierarchy. Any C children determined in Step 2 will continue to be children. Note that a given day may have multiple clusters in it. This approach is used in our prototype, and is useful for creating summaries that are intended to be navigated using time-based interfaces, as is the case with the Calendar Browser.

An alternative is to create the higher levels of the hierarchy using statistical techniques analogous to the ones of Step 2. For example, instead of analyzing the time between photographs as we did in Step 2, we could analyze the time between clusters. Clusters that are relatively closer together to each other than other groups of clusters could then be combined into parent nodes at a higher, possibly newly created, tier in the cluster tree. This process would be repeated recursively, building the tree upwards from the initial clusters.

3.2. Creating Summaries

Collections of personal photographs typically contain many more photographs than can be displayed on screen at once. This problem applies not only to desktop photo browsers, but even more so to low resolution devices such as televisions, hand-held computers, and displays on digital cameras. Most systems overcome this limitation by allowing users to scroll through sets of images. Unfortunately, scrolling also has limitations. Most notably, scrolling gives no sense of overview, making it easy for users to feel lost in a sea of images.

An alternative to scrolling is summarization, as in the display panel of Figures 1a and 1b. Instead of displaying all images, a set of representative images is shown. Once the user finds an image from the event he is looking for, he is able to "zoom in" and see the event in greater detail. When he is finished examining the event, he may "zoom out" to return to the summary view. Detail is hidden when unnecessary, but available whenever needed. Overviews are easily seen by zooming out.

The input to our summarization procedure is a set of sequential clusters **C** at level *k* in the hierarchy, plus a target *T*, the desired number of representative photographs. With the Cluster Browser, summarization always starts with a single day, month or year cluster, but in general we can summarize a set of consecutive clusters. The following two steps perform the recursive summarization process:

Step 1: Screen space assignment

If there are *T* or fewer photographs in **C**, creating a summary is trivial — we assign screen space to each of the images. Otherwise, we use the following rules to assign screen space:

1. Assign one space to each cluster in **C**. This ensures that there will be at least one photograph from each cluster in the summary. If we are unable to give one space to every cluster, we give priority to larger clusters.

2. Say we assigned *M* spaces in Step 1. We then assign the remaining $T - M$ spaces to clusters in proportion to their sizes.

For example, say we are summarizing three clusters. Cluster C_1 (including its children) has 20 photos, C_2 has 10 and C_3 has 5. Our target is 10 summary photos. After we assign one slot for each cluster, we are left with 7 spaces, so we give 4 more to C_1, 2 to C_2 and one to C_3. Thus, C_1 gets a total of 5 spaces, C_2 gets 3, and C_3 gets 2.

These rules are based upon the notion that summaries should show as much variety as possible within the given time range. This is achieved by giving summary space to as many clusters as possible, regardless of cluster size. For instance, a given time range may contain two bursts: a vacation when 100 photographs were taken, and an old friend's visit when two photographs were taken. If we

simply assigned screen space based on cluster size, a summary of ten images would contain only images from the vacation. Images from different clusters tend to be more distinct than images within a single cluster. Thus, by prioritizing summary images from different clusters, we ensure more variety.

Step 2: Selection of summarization photographs

As a result of Step 1, each cluster C_i in **C** has been assigned a target number of photos T_i. If C_i is not a leaf cluster, then we recursively repeat Step 1 to allocate the T_i spaces to C_i's children. (Recall that if C_i has children, only its children have actual photographs.) Eventually we obtain a number of spaces assigned to each leaf cluster in **C**.

There are several options for choosing representative images in leaf clusters:

- *photographs separated by smallest difference in time* - Images with little time between one another are likely to be of the same subject. Any event that warrants multiple photographs is significant enough to be considered representative. Thus, one of the photographs of the sequence would be a good candidate for the summary.
- *photographs separated by largest difference in time* - Images with lots of time between one another may visually differ greatly from one another. Thus, the photographs right before or after the long time interval may be of interest.
- *contrast and resolution information within photographs* - In some cases, choosing a representative image is not as important as choosing a highly visible image. Images with high contrast and resolution tend to be easier to identify within summaries.
- *cluster-wide image analysis* - Based on the image properties of the cluster, it may be good to include an image that best represents the visual characteristics of the cluster.

In our prototype, we use the first two heuristics only. First, we look for the smallest time difference between consecutive photographs, and we select one of the two involved. If we still need additional summary photographs for the leaf cluster, we look at the largest time differences and select the photographs involved. For example, say we need 4 summary photos. If *a* and *b* are the closest photos, *b* is our first choice. Then say that *c*, *d* is the pair with the largest difference, and that *e*, *f* follows. Our remaining choices would be *c*, *d*, and either *e* or *f*. In practice, we have found that the actual details of how photographs are selected are not critical, as long as a reasonable strategy is used.

4. OTHER BROWSERS

As we informally tested our Calendar Browser on various users, we found that they liked the time-based approach and the summarizations, but they were in disagreement on the user interface. Some liked the browser's simplicity, others suggested that we borrow the explicitly hierarchical browsing controls of the Windows File Explorer and Macintosh Finder utilities. Figure 5 shows the resulting alternative browser, which we also implemented.

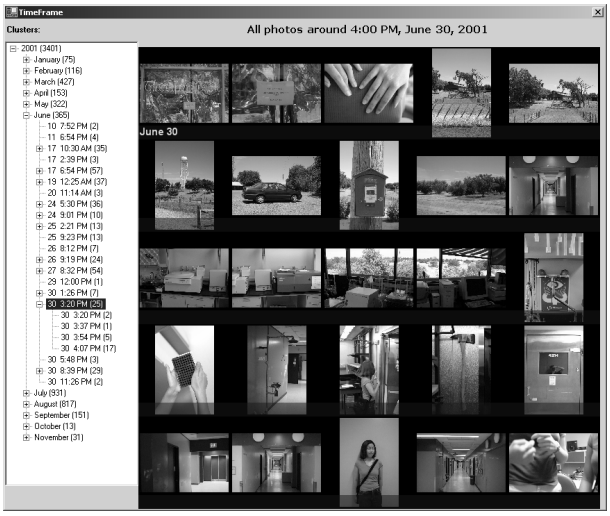

Figure 5: The Hierarchical Browser

The left panel contains a tree widget whose nodes at its different levels correspond to times at a given granularity. Outermost nodes correspond to years. Once a year is opened up by clicking on the square next to the year's label, months are shown indented. Users may open and close nodes at various levels, just as they do when browsing a hierarchical file structure. The numbers in parentheses next to each node label are the numbers of images nested below that node. For example, Figure 5 shows that on June 30, 2001, 25 photos were taken between 3:20pm and 5:48pm.

The number of entries on each level depend on the results of the clustering process. For example, while all the images taken on June 20 are combined in a single entry, June 30 ended up containing five sub-clusters, each being a burst of picture-taking.

The right pane is the same as the display panel of the Calendar Browser, except that clicking on an image has no effect. The functionality of 'drilling down' is instead covered in the tree control widget. Note that, unlike the Calendar Browser, the Hierarchical Browser allows one to drill down to sub-clusters smaller than a day.

These two browsers are rather different, and before proceeding, we needed to understand the characteristics of their design components. In addition, we were curious how well either design held up against an existing, commercial photo browser. In particular, we were interested in the following questions:

- How quickly can users find a given image, especially when multiple similar images are also part of the collection?

Figure 6: ACDSee Commercial Browser

- How effectively will users find images when given a description, such as "mountains with a sunset", "a red telephone", "a woman in a blue T-shirt?"
- Will the time-based summaries facilitate searches for time-related images, e.g., a Halloween or a Christmas picture?
- Do users spontaneously take advantage of the summarizations and time-based organization?
- How good is the summarization when compared to random selections of images?
- Are there browser-related differences among computer novices and experts?

We therefore conducted a controlled study covering our Calendar Browser, our Hierarchical Browser, and the commercial ACDSee browser. Figure 6 shows a screenshot of ACDSee.

This browser arranges images in a linear order. Users scroll through the photos vertically with a standard scroll bar. ACDSee also features an integrated file browser attached to the left of the image pane (not shown in the Figure). This browser is useful when images are already manually clustered in sets, and placed in different folders of the file system. The browser also allows drag-drop between the image pane and folders. We disabled these facilities for our experiments because we are currently interested only in evaluating browsing without any manual organization efforts on the user's part. We did order the images in ACDSee chronologically. Subjects could find the date and time of each image by briefly hovering the mouse pointer over the image.

Both of our browsers are particularly effective for users who originally took the photographs they are exploring. In the case of photographic Personal Digital Libraries, it is very common for the curator also to be the creator of the collection. Nevertheless, as our user experiments show, even strangers to a collection benefit greatly from these browsers.

5. EXPERIMENTS

We tested 12 subjects, six computer 'novices', and six 'experts'. We defined a novice as someone who uses computers at most to read email, create text documents, and browse the Web. If subjects also used other applications or programmed, they were classified as experts. Ages ranged from 18 to 55+. Among the twelve subjects, six were male, six female. Professions included students, school teachers, and university professors of varying disciplines, other than Computer Science.

We videotaped and recorded all test sessions. The sequence of the subjects' exposure to the three browsers throughout the experiment was balanced to neutralize learning effects across browsers. We timed all interactions with the browsers.

We used two image sets, A and B. Set A contained 1000 images, set B contained 3500. All photographs were in color and taken in a variety of locations, seasons, and times of day.

All values reported in this paper vary in their margins of error between ±5% and ±23% at 85% confidence level. Unless otherwise stated, all results reported below are statistically significant at 85% confidence level.

We asked subjects to complete a variety of tasks in six different categories, within a maximum time limit. Tasks in category 0 involved dataset A, all other tasks operated on dataset B. In what follows we describe the task categories, and as we do so, we describe the results obtained.

Category 0: Find a given image in dataset A. Time limit: 5min.

Category 1: Find a given image in dataset B, given also the month during which the image was taken. Time limit: 2.5min.

Subjects did a total of 9 tasks: three of category 0, each with a different browser, and 6 of category 1. For the latter

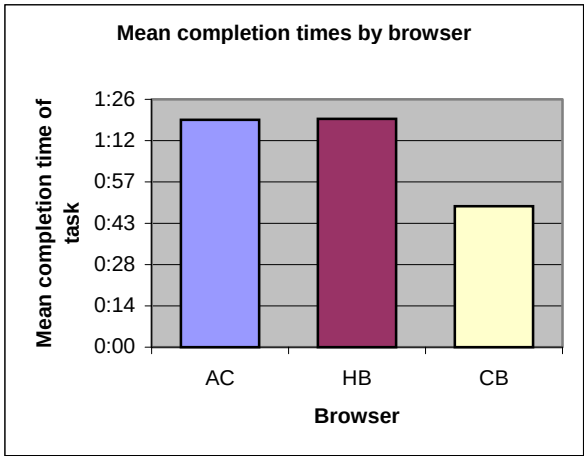

Figure 7: Average Completion Times for Tasks of Categories 0 and 1

6, each browser was used with two separate images. Again, the order in which subjects used the browsers was varied to eliminate biases.

Figure 7 shows average completion times for the combined Tasks 0 and 1, for the cases where the subjects did complete the tasks. While browser ACDSee (AC) and the Hierarchical Browser (HB) draw just about equal, around 1 minute and 20 seconds, the Calendar Browser (CB) enabled significantly faster completion times at an average of about 50 seconds; an average 33% improvement. In our test runs, this advantage of CB over the other browsers held for computer experts and novices alike. However, we have too few data points in each of these two groups separately to make a general statement on this novice/expert point.

In some cases the subjects did not find the requested images before the time limit expired. Figure 8 shows a comparison of tasks unable to be completed across browsers. ACDSee did by far the worst, with 31% of the tasks unable to be completed. The Hierarchical Browser was the most reliable for finding the images (6% incomplete). Users of the Calendar Browser were unable to complete tasks around 11% of the time. The difference in completion success between the Hierarchical Browser and the Calendar Browser are not significant.

The results of Figures 7 and 8 show that the time-cognizant browsers, which perform summarization, can indeed help when subjects are searching for images. Some of the improvements are due to the time-based navigation scheme, but other gains are due to the summarization capabilities that let the subjects get meaningful overviews. These results also indicate that although users were exposed to new browsing interfaces, they were able to understand them well enough with minimal training to perform at levels at least as good as with a traditional browser.

Category 2: Given 10 textual descriptions of images, find as many images that fit any of these descriptions as possible. Time limit: 3min.

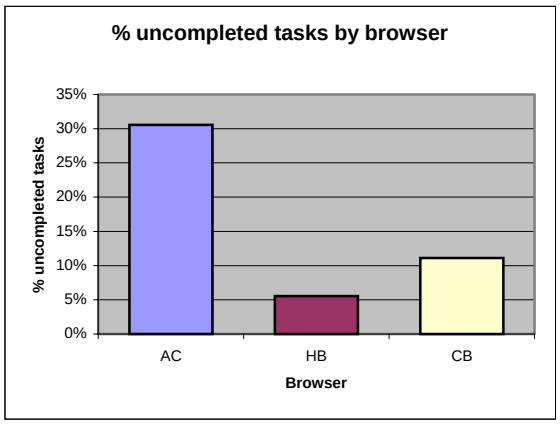

Figure 8: Average Completion Success for Tasks of Categories 0 and 1

Category 2 was designed to measure performance when classes of images were desired, not just a particular image. To generate the 10 descriptions for a category 2 task, we selected at random 10 photographs from the collection. For each image, we then manually wrote a description. For example, the resulting descriptions could be "a sandy beach," or "a man sitting at a bench." Each subject did two tasks of Category 2, with different description sets, one time with CB, the other with ACDSee.

Our results for Category 2 indicate that, when confronted with textual descriptions of images, subjects performed at identical levels in the linear ACDSee as in the Calendar Browser. Nearly everyone used linear search even with the Calendar Browser.

Category 3: Find a given image, in the case when a similar image appears in one of the month summaries. In addition to completion time, we noted whether subjects made use of the summarizations, or followed another strategy (usually linear search). Time limit: 5min.

Each query image was selected as follows. A random photo was selected from one of the 12 month summaries. Then a similar (but not identical) photo was selected from the same leaf cluster. The subject was given the similar photo as the query, not knowing how it was selected. Subjects performed this task with both CB and HB, with different query images.

Category 3 measured the impact of subjects ignoring vs. taking advantage of summarization when looking for a given image, as opposed to working from textual descriptions. In Task 3, 42% of subjects made use of summarization (across both browsers). That is, these subjects browsed 'horizontally' through time; when they found an image similar to the sample they had been asked to find, they drilled down. Figure 9 shows that these 42%

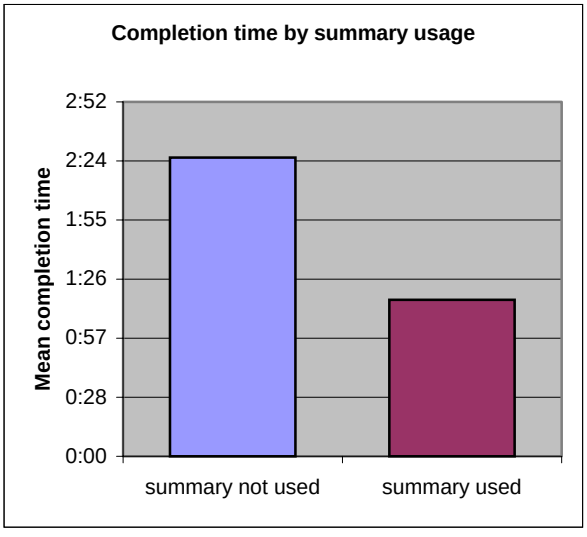

Figure 9: The Use of Summarization

who made use of summarization enjoyed a 48% boost in performance.

All of the summary users finished tasks of Category 3 within the allotted time, while 36% of subjects who did not take advantage of summarization were unable to complete the task

Category 4: Find a given image, when the image has a clear clue as to which time of year it was taken. For example: a Christmas tree, fire works, etc. When appropriate, we ensured during the debriefing sessions at the end of the experiment that no cultural differences had prevented a subject from making the time association. Analogous to Category 3, we noted whether subjects recognized and used the time clue. Time limit: 5min.

Category 4 allowed us to examine whether subjects made use of time clues in images, and how this use affects performance. Again, subjects used CB and HB.

Almost all subjects, 92%, made use of the time clues for these tasks. Everyone completed the task. The average completion time was 36 seconds. The 8% who did not use the time clue averaged 1min 18sec. However, 8% of our subject population is too small a data set to make this performance comparison statistically significant.

Category 5: Find images to fit 10 textual descriptions of leaf clusters. Time limit: 5min for each run.

To generate the descriptions, we manually clustered the collection by event, and gave each cluster a short description (e.g., “visit to San Francisco”). For each task instance we selected 10 descriptions at random (no replacement). We limited subjects to the 'year' and 'month' summary views in CB.

Each subject performed two category 5 tasks, using the Calendar Browser each time. For one of the tasks, however, CB was altered so it would select images at random from the proper time span. For the other task, images for the summary were selected as described earlier.

Tasks in category 5 forced users to operate in summary mode, allowing us to examine whether our summarization made any difference when compared to random excerpting from the collection. We found that even at a confidence level of 99% there was a significant difference between random and clustered summarization. We measured a 56% performance improvement.

5.2. Discussion

We are very encouraged by our findings. These were intentionally tough experiments in that we included subjects who were not deeply familiar with computers and, most importantly, were unfamiliar with the collection. In our main target application of Personal Digital Libraries users will know the collection, so summaries and time organization will have even greater positive impact on these users.

We are also delighted by the positive results, because the systems we report on here operate under the most demanding assumption that users invest absolutely no organizing effort. We can expect that even small user efforts, such as high-level captioning will improve upon these results even further.

However, users did not always take advantage of the summarization capabilities. Overall, only 42% of users took advantage of summarization in our Hierarchical and Calendar Browsers. This tendency to overlook the summarization feature was particularly evident as subjects needed to overcome the barrier of abstraction that we erected by the verbal nature of tasks of category 2 (finding as many images as possible from 10 textual descriptions). In that case, subjects were also strained by trying to keep multiple image descriptions in mind simultaneously. This strain may have contributed to that choice of the simplest, linear, search strategy.

The fact that 92% of the subjects had a specific time association with certain photographs (Category 5) underscores the importance of making time visible in the interface. Most browsers do not do this; PhotoMesa and the two browsers we developed are notable exceptions [9].

The ingenuity of many subjects impressed us. Some novices used binary search algorithms. Some used clothing of figures in images for clues on season or occasion. For example, one subject explained how she proceeded in one instance, using the Hierarchical Browser: "I used the time for the sunset images because I knew I was looking for something late in the day." It is seductive to use some of these observations as the basis for adding search and browse features. However, it is very difficult to keep the interface simple enough that computer novice and expert users can operate the systems with virtually no instruction.

During the debriefing sessions, subjects were clear in their evaluation of the ACDSee browser: “It seemed like I did a lot of scrolling, when there should have been a more efficient way [to navigate the images].” “The linear one [AC] was more frustrating than the others. You literally felt like you were looking for a needle in a haystack.”

Opinions about the Hierarchical vs. the Calendar Browser were divided. Praising HB: “I like having the orientation of some sort of text on screen;” and “Having the knowledge of how many [photos were in a cluster] was helpful;” or “I liked the one [browser] with the tree. It was the easiest to navigate. You could skip around all at once without having to go through the different levels.” On the other hand, other subjects preferred the Calendar Browser: “I'd rather click on a picture than a month [label];” and “The zoom-out zoom-in was more intuitive [than HB]. You didn't have to think about it as much.” Although for testing purposes we kept the HB and CB interfaces separate, we plan to integrate these interfaces in the future, based on the positive response we received fromxs users in both cases.

6. CONCLUSION

We designed, implemented, and tested two photo browsers that can accommodate thousands of images. These systems

are intended primarily for personal digital libraries. Both browsers are built atop a common architecture. This architecture consists of metadata extraction, cluster engine, and summarization modules that are replaceable.

Our browsers use cluster analysis of the times when photographs were taken as the foundation for summarizing large subsets of the images within a collection. The browsers differ in how users navigate the collections.

We conducted user studies that compared various design aspects of the two browsers against a commercial image browser. We found that our Calendar Browser, which affords time navigation through direct-manipulation, enabled a 33% improvement in speed of finding given images in collections. We found that our summarizations improve the search for photographs from textual descriptions by 56% over systems that fill the screen with randomly selected excerpts from the collection.

While summarization clearly improved performance, only 42% of subjects thought of making use of the advantages that summarization provided. We plan to address this issue in subsequent designs.

In contrast, subjects readily recognized and used our time-based organization. While subjects clearly preferred this organization to an alternative linear solution, they were divided over which other user interface is most convenient for navigating the photographs through the time space.

Our experiments show that we will be able to make Personal Digital Libraries of digital images manageable and convenient to browse. Digital cameras are rapidly creating large professional and personal image collections. Without substantial support from Digital Library technologies, this exciting new opportunity will be wasted, and turn into the digital version of the unorganized shoebox of photographs in a closet. With the proper tools, on the other hand, the records of individual histories will be fun and instructive to maintain and explore.

Acknowledgments

We are deeply appreciative of our subjects, who mostly live packed lives and have no need to sit through photo browser experiments to fill their days. The many, many minutes and seconds they spent with us turned into all the wonderful data points that make this paper possible.

REFERENCES

[1] Scott Stevens, Michael Christen, and Howard Wactlar. Informedia: Improving Access to Digital Video. *Interactions*, 1(4):67–71, October, 1994.

[2] Kobus Barnard and David Forsyth. Exploiting Image Semantics for Picture Libraries. In *Proceedings of the First ACM/IEEE-CS Joint Conference on Digital Libraries*. ACM Press, 2001.

[3] J. Frew, M. Aurand, B. Buttenfield, L. Carver, P. Chang, R. Ellis, C. Fischer, M. Gardner, M. Goodchild, G. Hajic, M. Larsgaard, K. Park, M. Probert, T. Smith, and Q. Zheng. The Alexandria Rapid Prototype: Building A Digital Library for Spatial Information. In *Advances in Digital Libraries '95*, 1995.

[4] Microsoft Windows XP. 2001. Information at http://www.microsoft.com/windowsxp/default.asp.

[5] Apple Macintosh OS X. Information at http://www.apple.com/macosx/.

[6] ACD Systems ACDSee Browser. Available at http://www.acdsystems.com/English/Products/ImagingProducts/ACDSee/ACDSee/index.htm.

[7] H. Kang and B. Shneiderman. Visualization Methods for Personal Photo Collections Browsing and Searching in the PhotoFinder. In *Proceedings of IEEE International Conference on Multimedia and Expo (ICME2000)*, pp. 1539-1542. IEEE, New York, 2000. Available at http://www.cs.umd.edu/hcil/photolib/paper/ICME2000-final.doc.

[8] Allan Kuchinsky, Celine Pering, Michael L. Creech Dennis Freeze, Bill Serra, and Jacek Gwizdka. FotoFile: a consumer multimedia organization and retrieval system. In *Proceedings of the Conference on Human Factors in Computing Systems CHI'99*, pp. 496-503, 1999. Available at http://www.acm.org/pubs/citations/proceedings/chi/302979/p496-kuchinsky/.

[9] Benjamin B. Bederson. *Quantum Treemaps and Bubblemaps for a Zoomable Image Browser*. Number HCIL 2001-10. University of Maryland, College Park, 2001. Available at ftp://ftp.cs.umd.edu/pub/hcil/Reports-Abstracts-Bibliography/2001-10html/2001-10.pdf.

[10] Apple iPhoto. 2002. Information at http://www.apple.com/iphoto/.

[11] John C. Platt. AutoAlbum: Clustering Digital Photographs Using Probabilistic Model Merging. In *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries 2000*, pp. 96-100, 2000.