Automatic Organization for Digital Photographs with Geographic Coordinates

Mor Naaman, Yee Jiun Song, Andreas Paepcke, Hector Garcia-Molina
Stanford University
{mor, yeejiun, paepcke, hector}@cs.stanford.edu

ABSTRACT

We describe PhotoCompas, a system that utilizes the time and location information embedded in digital photographs to automatically organize a personal photo collection. PhotoCompas produces browseable location and event hierarchies for the collection. This organization is created using algorithms that interleave time and location to produce an organization that mimics the way people think about their photo collections. In addition, our algorithm annotates the generated hierarchy with geographical names. We tested our approach on several real-world collections and verified that the results are meaningful and useful for the collection owners.

Categories and Subject Descriptors

H.3.7 [**Information Systems Applications**]: Information Storage and Retrieval—*Digital Libraries*; H.5.1 [**Information Systems Applications**]: Information Interfaces and Presentation—*Multimedia Information Systems*

General Terms

Human Factors, Algorithms

Keywords

Photo browser, geo-referenced photos, GPS, personal photo collection

1. INTRODUCTION

Location is one of the strongest memory cues when people are recalling past events [18]. Lately, technology advancements made it feasible to add location information to digital photographs, namely the exact coordinates where each photo was taken[1]. The location information can be extremely helpful in organizing and presenting image collections, starting from a global library of digital photos [17] to a single user's personal collection [16, 17]. In previous work [11], for example, we have shown how location information can serve as a basis for sharing photo captions between cooperating users who take geo-referenced digital photographs.

[1]While location-aware cameras may become a standard in the future, there are a number of ways to produce "geo-referenced photos" using today's technology. For a summary, see [17].

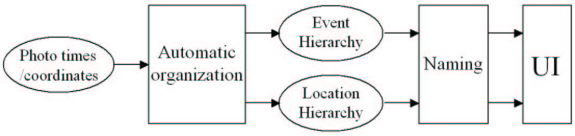

Figure 1: PhotoCompas system diagram.

In this paper we address the problem of automatically organizing a personal geo-referenced photo collection, in order to facilitate efficient search and browsing for specific photos, or for photos of particular events.

In particular, we are looking to automatically generate a structure that will enable browsing of the collection *without* the use of a map. Maps are extremely wasteful in screen real estate. In many cases, especially in personal photo collections, pictures may be sporadic in one location, and highly concentrated in another location. Having to pan and zoom a map to the correct low-level location may be cumbersome. This problem intensifies when the user operates on a small-screen device. In addition, many people do not feel comfortable using a map - in a computerized or even a non-computerized environment.

Our system, PhotoCompas, performs two major tasks. First, it automatically groups the photos into distinct events and geographical locations. Second, PhotoCompas suggests intuitive geographical names for the resulting groups. Figure 1 illustrates the processing steps and outputs of the system.

To create the event/location grouping we use a combination of existing time-based event detection techniques [7, 8] and a temporal-geographical clustering algorithm [5] to group photos according to locations and time-based events. This paper presents, to our knowledge, the first published algorithm that proposes, implements and experiments with an event-detection algorithm that uses location information in addition to time. Moreover, the location-based grouping enables a new way to access a photo collection.

For the second task, PhotoCompas generates a textual caption that describes, in geographic terms, each location or event. It is crucial to generate a good set of captions as we are not using a map to identify geographic location to the user. Our technique utilizes a geographical dataset of administrative areas (provinces, cities, parks, counties etc.) and a web search engine such as Google [6] to generate these captions.

In this paper we explore and experiment with the PhotoCompas set of algorithms. We also construct a user interface that uses the algorithms output. For reasons of space and a the currently-limited subject pool we will report on the user interface aspect of our work separately.

In Section 3 we describe the algorithms we use to discover the inherent structure of a user's photo collection. Section 4 shows how we generate a human-readable geographical name for a set of photos. In Section 5 we evaluate PhotoCompas by running experiments on three large sample collections of personal geo-referenced photos.

2. RELATED WORK

Automatically detecting events in a photo collection is a difficult problem that has been well studied in recent years [2, 4, 7, 10, 13]. All these event-detection techniques are based on the times the photos were taken, and some also augment the time data with information about the visual similarities of the photographs. While some (e.g., [2]) propose that their algorithms can be extended to support location information, such a study was not published yet.

There is, of course, an abundance of time-series clustering algorithms that can be applied to geographic data. For references to some of them, see [5]. In addition, there are many clustering algorithms and techniques that do not necessarily treat the data as a series (e.g., k-Means); while it is possible to use these for our purpose, we claim below that it is useful to utilize the time series data and the time values for the geographic clustering.

A number of technologies may assist in the task of naming geographical location in the future. A location-aware information retrieval engine, as proposed by Jones et al in [9] may provide a way to perform this task. It is not clear when such a search engine will be practical. In [11] we propose a system that enables sharing of photo labels between different photographers, using a digital camera with location capabilities. We showed how the system, named LOCALE, can be used to identify landmarks that appear in the photos. In addition, our system can be used to identify possible names for larger regions, based on the geographical spread and term frequency of words that appear in photo captions. Unfortunately, LOCALE requires a critical mass of caption submissions for geo-referenced photos, which is not available today.

3. DISCOVERING THE STRUCTURE OF AN IMAGE COLLECTION

The goal of our automatic processing is to group the user's photos using two different categories, corresponding to the natural way users think about their photos [3, 14]. The first category is location, and the second is event. That is, we wish to group the photos into hierarchies of locations and time-based events. Naturally, these two dimensions interact: photos from a certain event are associated with a location; any location may have pictures taken in it at different times (for example, multiple trips to Yosemite National Park).

Note that both time and location can easily be subjected to some pre-defined hierarchy. For example, any specific time can be easily categorized into a year, month, day, hour etc. Similarly, a location can be categorized by country, state, county, city etc. On the other hand, both time and location data can be grouped using only their continuous values, without adhering to any imposed hierarchy.

The main argument against using a pre-defined hierarchy is that it is often too rigid or not well perceived by users. A few examples that illustrate the problem may be an event that crossed day/month/year boundaries; proximate photos taken across state or city boundaries. Another problem is the hierarchy may be too bushy (many different cities in one state, or picture-taking days in one month). In addition, as we show in the next subsection, our generated hierarchies inform each other in a way that is not feasible with pre-defined hierarchies.

Instead of using a pre-defined hierarchy, we automatically create a special hierarchy for each collection in the two categories, using the values for time and location of the photos as real values in a continuous space. To refer to groups of photos in these hierarchies, we define two terms below, to be used throughout this paper. Both terms will be defined more precisely in Section 3.2:

A cluster is a node in the location hierarchy, a group of photos that belong to one geographic location.

A segment is a *sequential* group of photos. In particular, segments can represent user events.

Of course, in order to display the event and location information to a user we must use terms from well-known hierarchies (e.g., year, month, city names, ...). For example, naming a location "Cluster Number 4" is of little value. However, naming it "Boston, Massachusetts" is meaningful, even if the name may not be a strict description of the cluster contents. However, as shown in Figure 1, we only worry about naming the generated clusters and segments *after* the structure for the collection has been generated. In fact, we may merge (as a final step) clusters that occur in the same city - simply because we have no means of making them distinct to the user.

Eventually, PhotoCompas generates two distinct hierarchies for the user interface, location and event. However, the user interface allows filtering based on both hierarchies at the same time, so the user is able to click through any "virtual" path that interleaves locations and events.

3.1 Output Requirements

As Figure 1 shows, the output of the automatic organization step are location and event hierarchies. Before we describe

the details of the algorithm in Section 3.2, we list the requirements and guidelines for each of the outputs.

3.1.1 Requirements for Event Category

People often think of their photos in terms of events: consecutive photos corresponding to a certain loosely defined theme such as a wedding; a vacation; a birthday etc. [3, 14]. Users inspecting their own photo collection ordered by time can easily draw boundaries between the different photographed events. We wish to mimic this human inspection as accurately as possible.

We make use of the "Story Line" assumption: all photos are taken by a single photographer, or alternatively, using a single camera (used by a number of family members). We make this assumption so we can treat the photo collection as a sequence of photos, coherent in space and time (i.e., no two pictures are taken at the same time in two different places). This assumption is not as restricting as it seems when moving on to a family collection with a few contributors and possibly a few cameras. Modern digital cameras insert the camera make and model as part of the photo metadata. If more than one camera appears in the collection, we can use this information together with time/space filtering to separate the different cameras and treat the photos as two separate sequences.

A second observation, verified in a number of publications [2, 4, 7], suggests that people tend to take personal photographs in bursts. For instance, lots of pictures might be taken at a birthday party, but few, if any, pictures may be taken until another significant event takes place. We take advantage of this burstiness in discovering the event structure of the collection.

The event category can be flat or hierarchical. A flat category means we only identify the top-level events - in other words, the points in the stream of consecutive photos where the context has changed (e.g., "A birthday party", "Trip to Mongolia", "4th of July"). In this paper we only create top-level events as they are the most crucial for browsing.

Our event categorization follows one strict rule:

i. ("Story Line") Only consecutive photos can belong to the same event ε. $p_1, p_2 \in \varepsilon \wedge (time(p_1) < time(p_3) < time(p_2)) \Rightarrow p_3 \in \varepsilon$.

In addition, a number of (sometimes conflicting) observations serve as guidelines for our processing algorithm:

ii. A gap of h hours between consecutive photos is often an indication for a new event. The value of h should change dynamically as informed by knowledge about the geographical clusters: more popular location means lower h value (event granularity is lower, for example, around the user's hometown).

iii. Similarly ("Burst" assumption): within one event, photos are taken at a steady rate; an unusual time gap between photos emphmay signal the beginning of a new event.

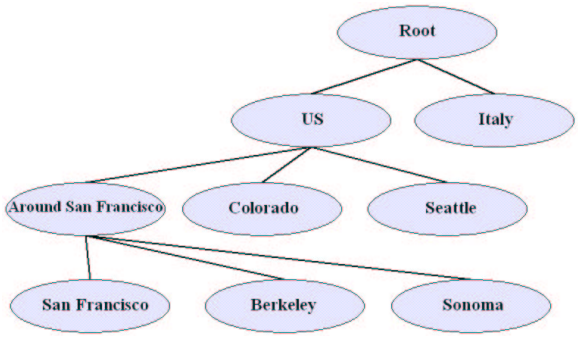

Figure 2: A sample location hierarchy of a collection, using textual names to illustrate the clusters.

iv. The physical distance between the locations of two consecutive photos is another possible predictor for event boundaries.

3.1.2 Requirements for Location Category

In this section we present the requirements and guidelines we use for the location hierarchy. The location hierarchy is used both for presentation in the user interface, and to inform the event segmentation.

Ideally, we would like users to be able to quickly "drill into" one specific location where they had taken photos. Our location hierarchy could in principle be arbitrarily deep, but our implementation uses a 2- or 3-level hierarchy.

The first level of the hierarchy is pre-defined: the country level. We use the pre-defined country classification since the world's division into countries is (generally speaking) well defined, and users are well aware of the "country" context: most people always know what country they are currently taking photos in. Therefore, we initially split all photos based on the country in which they were taken.

The second level of the hierarchy, as discussed above, is created by grouping the photos into *clusters* that are expected to make sense to the user. These clusters may vary, both in terms of the size of the area they represent, and the number of pictures in each cluster.

Figure 2 shows an example of such a location hierarchy, using textual names to illustrate the *general area* each cluster represents. This hierarchy is in fact part of the hierarchy created for one of our test collections, Z (see Section 5).

The clusters can be recursively split into lower-level clusters (like the "Around San Francisco" cluster in Figure 2). To decide when to split clusters, we use time information. We split a cluster when the number of occasions the user had visited this location exceeds a threshold. The intuition behind this criterium is that people are more familiar with areas where they have taken photos on many different occasions.

To summarize, we want to create a location hierarchy that correctly represents the unique locations the user had visited. Our hierarchy is created while following these rules:

v. The tree must not be too bushy: $|descendents(\ell)| < n$ for every location node ℓ. We used $n = 10$ to keep the user interface menu sizes reasonable.

vi. No redundant inner nodes with only one descendent: $\forall \ell : |descendents(\ell)| = 0 \vee |descendents(\ell)| > 1$.

Additionally, the hierarchy *tries* to follow these guidelines:

vii. Location nodes that represent very few photos are merged with other nodes, unless they show a substantial difference in their geographical location (based on distance and compared to distances between other clusters).

viii. At any level of the hierarchy, photo p belongs to the node whose center is geographically closest. $p \in \ell \Rightarrow \forall \ell_i \neq \ell : samelevel(\ell_i, \ell) \vee (dist(p, \ell) < dist(p, \ell_i))$ where the distance *dist* between a photo and a cluster is the distance from the photo to the cluster center.

ix. Photos taken in close time proximity should belong to the same location cluster. Here we again use the time data (given the Story Line and Burst assumptions) to inform our geographic hierarchy. This rule will prevent pictures from the same event to be categorized to two different locations, and moreover, is likely to have the benefit of keeping together related locations that otherwise may have been split.

x. Leaf clusters in the hierarchy are not overloaded, and should represent the level of knowledge the user is assumed to have of this location: if too many segments belong to a single leaf cluster, split this cluster into further geographic sub-clusters.

Often these guidelines can conflict with each other, or be impossible to adhere to given the user's particular set of photos. In the next subsection we present our algorithm that makes use of these guidelines while trying to balance the conflicting rules.

3.2 A Three-Pass Algorithm for Generating Location and Event Categorization

The algorithm described in this section creates both location and event hierarchies, and assigns all the photos to nodes in each.

As mentioned above, we assume that users are well aware of the countries where they are taking photos. Therefore, we process all photos from each country separately (we have a geographical dataset that enables us to query the location of each photo to find the country where it was taken). For the remainder of this section, we assume that all photos are taken in one country.

Rather than separating the time-based processing from the location-based processing, the implementation uses a hybrid time/space technique. The first step treats the photos as a sequence, and looks at the time and geographical distance between each pair of consecutive photos to create an initial grouping into *segments* representing low-level events. Then, we cluster these segments using a geographic clustering algorithm. Finally, we make a final time-based pass over the sequence of photos to decide the final breakdown into events

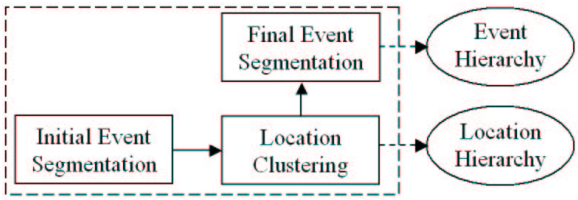

Figure 3: Processing steps in our automatic organization algorithm.

(informed by the newly created location clusters). The process is illustrated by Figure 3.

Here we define more precisely the two concepts we introduced earlier, a *segment* and a *cluster*. Let $P = (p_1, p_2, \ldots, p_n)$ be the set of user's photos, ordered by the time the photos were taken. A segment $S(i, j)$ is a set of consecutive photos $(p_i, p_{i+1}, \ldots, p_j)$. A *segmentation* M of P is defined by $k+1$ indices $(b_1, b_2, \ldots, b_k, b_{k+1})$ that divide P into k segments $S_i \equiv S(b_i, b_{i+1})$ with $1 = b_1 < b_2 < \cdots < b_{k+1} = n$.

A *cluster* C is a set of segments that fulfill some predicate *pred*: $C = \bigcup (S_i | pred(S_i) = true)$. Semantically, we use clusters to represent segments (and thereby, photos) that occur in the same location ($pred =$ "occurs in location X"). A cluster can include photos from non-sequential segments.

Segments are an approximation to user events, as they divide the photo collection in time. A perfect segmentation $M_{optimal}$ is the segmentation of P into ground-truth user events – a segmentation humans will create given their own photos. An *over segmentation* M_{over} of P is one where the set of indices is a superset of the indices in $M_{optimal}$.

Our three main steps of Figure 3 can therefore be expressed now in more accurate terms:

1. A linear pass over the sequence of photos, to generate an approximation to M_{over}.
2. A geographic clustering computation that takes M_{over} and generates a set of clusters C_i, each containing one or more segments from M_{over}; each segment belongs to exactly one cluster. Each cluster corresponds to a distinct geographic location.
3. Another linear pass over P, merging adjacent segments from M_{over} that are likely to be related based on the results of the geographic clustering. This stage results in a final segmentation M.

Thus, our algorithm produces a segmentation M and a set of clusters C, each containing one or more segments from M. At this point, the clusters C should represent the geographical locations where the user had taken photos. The segments $S_j \in M$ represent the different events in the user's photo collection.

We now present the algorithm in more detail, while highlighting the rules and guidelines that are aided by the different steps.

3.2.1 Step 1: Computing M_{over}

To approximate M_{over}, we use a variation of our segmentation algorithm presented in [7]. This algorithm is based on the "bursty" nature of photo collection, and is performed in two linear passes. On the first pass, we iterate through the photos in P - recall that they are ordered by time (rule i). During this pass, the index $i+1$ is added to a segmentation M every time two consecutive photographs p_i, p_{i+1} differ by more than a specified time h_{over} (guideline ii). Our earlier experiments have shown that the algorithm is not very sensitive to the chosen value of h_{over} when it is between 6-24 hours; we chose $h_{over} = 12$.

In the second pass, these initial segments are split into finer segments based on the time differences *and distance* between photographs within each initial segment (guidelines iii, iv). The splitting is done by computing the time difference and geographical distance between all consecutive photos in the segment. We then scan this list of differences, and look for outlier values using well-known statistical methods (described in [7]). We split the segment at a point where we find a time difference outlier and a distance outlier at the same point (i.e., between the same two photos). A split between photos (p_j, p_{j+1}) means adding the index $j+1$ to the segmentation M_{over}. The statistical thresholds are conservatively set such that the sequence of photos P is over-segmented. Although it is not guaranteed that the final result, M_{over}, will be a superset of our "perfect segmentation" $M_{optimal}$, it is likely to be a close approximation to such a superset. We verify that this is indeed the case in Section 5.1.

3.2.2 Step 2: Creating the Location Clusters

Once we have created the segmentation M_{over}, the next step is to find a grouping based on location for photos in P. To this end, we use an algorithm we call *SegmentCluster*, described in [5]. We assume the photos in each segment occur in a specific geographic location (guidelines iv, ix). The problem solved by *SegmentCluster* can be defined as follows. Find an assignment of the segments in M_{over} to a set of location clusters C. Define *source locations* as the set of centers of these clusters, denoted by Γ. Given a cluster c, its source location $\gamma_c \in \Gamma$ and a segment $S_j \in M$, denote the likelihood that the source of S_j is γ_c (or, that S_j is assigned to c) by $Prob(S_j|\gamma_c)$. This probability computation takes into account the location of all the photos in S_j.

The goal is to find the clusters and the segments associated with each. Mathematically, we wish to pick k source locations $\gamma_1, \gamma_2, \ldots, \gamma_k$, and an assignment for each segment $S_j \in M_{over}$ to a source γ_{c_j}, such that the the total probability $\prod_{j=1}^{|M_{over}|} Prob(S_j|\gamma_{c_j})$ is maximized (guidelines $viii$, ix). The algorithm executes using multiple values of k, and applies the Bayesian Information Criterion (BIC) [15] to search for the value of k that is BIC-optimal (roughly speaking, BIC is maximizing the probability while keeping k as small as possible; this should provide for guidelines v and vii).

After the execution of *SegmentCluster*, we have a flat list of geographical clusters containing the photos in P. These clusters can differ in their area size, and the number of segments and photos associated with them. The third row in Figure 2 is an example (annotated by textual names) for such a list.

Often, some clusters will have too many segments associated with them, like the "Around San Francisco" cluster in Figure 2. As mentioned above, many segments associated with a cluster are an indication that the user is familiar with the geographical area. For each such cluster, we recursively execute *SegmentCluster* on the union of segments associated with the cluster. This step results in further breakdown of the location hierarchy as demonstrated by the bottom row in Figure 2, and helps follow guideline x.

An alternative to the *SegmentCluster* algorithm, which performs clustering based on groups of photos (the segments), is to simply cluster the set of photos P using some generic clustering algorithm such as k-Means. However, especially when clusters are not well separated, we find that such algorithm perform poorly, making rather arbitrary divisions of photos into clusters, and possibly violating guideline ix. On the other hand, *SegmentCluster* uses our additional knowledge of the segments (or events) in which the photos occur to make sure photos that belong together fall into the same geographical cluster (guideline ix). The cost of this policy is, possibly, a slight overlap in geographical coverage between clusters (violating guideline $viii$).

3.2.3 Step 3: Towards $M_{optimal}$

While our first goal of identifying the areas where photos occur is now complete, we do not yet have the grouping of photos into events. In step 1 we created an over-estimate of the event segmentation, M_{over}. In step 3 we try to obtain an approximation to $M_{optimal}$, the ground-truth of event segmentation – the way a user would have split her collection.

Step 3 is a linear pass over the photos in P, where we merge some adjacent segments in M_{over} that belong to the same cluster. In other words, if $S_i = (p_i, p_j)$, $S_{i+1} = (p_{j+1}, p_k)$ and $S_i, S_j \in c$, we remove $j+1$ from the segmentation M_{over}, creating one longer segment in place of two shorter ones. However, we merge the segments *only* if the adjacent segments are less than $h(c)$ hours apart, where $h(c)$ is an inverse function of the cluster's popularity - the more segments exist in this cluster, the smaller the value of $h(c)$. This follows the intuition of guideline ii, where less-visited clusters should be segmented more conservatively. In our implementation, we used $h(c) = Max(H_{min}, H_{max}/n^2)$ where n is the number of segments in that cluster within one year before or after S_i, and the values for H_{min} and H_{max} are set experimentally and are roughly a few hours and a few days, respectively (see Section 5.1).

At the end of step 3, we have a location cluster hierarchy, and an event segmentation M, which is hopefully a good approximation of $M_{optimal}$. We verify these results on sample collections in Section 5. In the next section we assign geographical names to the different clusters and events, so they can be presented in a user interface.

4. NAMING GEOGRAPHICAL LOCATIONS

After grouping the photos into event segments and location clusters, we need a way to present these results to users. As mentioned earlier, we are interested in investigating whether users can efficiently navigate the hierarchy without the use of a map. To facilitate this, we need to name each group of photos; i.e., give it some textual caption. We want to

give a geographical name to both the location clusters and the different events, as the latter, naturally, also occur in some geographic location. An event's location is often more specific than a location cluster. For example, we may have a location cluster with photos from the Silicon Valley; one of the detected events may be associated with the Silicon Valley cluster, but in fact had occurred in Stanford, and therefore can be described using a more precise geographical name. Practically, the names are required to be:

i. Informative and accurate, providing users with a good idea of the location they describe.
ii. Recognizable. Regardless of how accurate or informative a name is, it is of no use to a user that does not recognize it. In implementation, for example, we did not use county names as people often do not recognize them.
iii. Unique. No two sets of photos should have the same name unless they represent the same location (this can happen only when we are assigning a geographical name to events).
iv. As short as possible, to avoid clutter on the user interface, and allow users to quickly scan a list of names.

In the following subsections, we first deal with the general problem of generating a descriptive caption for a set of geographic coordinates. We then show how these techniques can be applied in the context of the event and location hierarchies that our system creates.

4.1 Naming a set of Geographic Coordinates

This section describes our approach to finding "good name" candidates for a set of geographic coordinates.

Our naming process has three steps. In the first step, we find for each latitude/longitude pair[2] the state, and city and/or park that contain it. This is done using an off-the-shelf geographic dataset of administrative regions. For example, a particular coordinate may be inside in California (state), San Francisco (city), and Golden Gate National Recreational Area (park). Another coordinate may be taken in Washington (state) and Seattle (city), but not in any park.

We count the frequency in which each city and park occur in the set of photo coordinates, building a term-frequency table. We weigh each type differently, with national parks weighed more heavily than cities; and cities weighed more heavily than other parks such as state parks or national forests, for example. The different weights allow us to give more importance to names that are more likely to be recognizable to users. At the end of this process, we have a *containment table* with terms and their score.

In the second step, we look for neighboring cities. By locating cities that are close to the coordinates in this set, and computing the distance from the center of the set to the city, we are able to produce textual names for these clusters such as "40 KMs south of San Francisco". We pick neighboring cities based on their "gravity": a combination of population size, the city's "Google count", and (inversely) the city's distance from the center of the set of photos. The "Google count" of a city is the number of results that are returned by Google [6] when the name of the city (together with the state) is used as a search term. We use this as a measure of how well known a city is, and thus, how useful it would be as a reference point. For example, a set of pictures taken at Stanford may be captioned "40 KMs South of San Francisco", or "30 KMs North of San Jose". The population of these cities is comparable, but since the Google count for San Francisco is much higher than San Jose, the former is chosen despite being further away. This step create a *nearby-cities table*, again with terms and their scores.

The final step involves picking 1-3 terms from the tables to appear in the text caption of each set of photo coordinates. For example, a possible caption can include the two top terms from the containment table, and the top nearby city: "Stanford, Butano State Park, 40 KMs S of San Francisco, CA". Our method of picking the final terms varies according to the semantics of the set of photos we are trying to name, as we explain in the next subsection.

4.2 Naming Location Clusters and Events

Picking the final terms to appear in the caption is done in different ways depending on whether we are naming an event from the event hierarchy; a leaf cluster in the location hierarchy; or an inner node cluster in the location hierarchy.

Naming a leaf cluster in the location hierarchy, such as the Berkeley cluster in Figure 2, is the hardest and most important of the three different types. It must be the most accurate description since there is no lower level where more location details are exposed. In addition, the location may have been visited a number of times, and may represent a heterogenous set of events occurring in slightly different locations. The rules for naming a leaf cluster are as follows: 1. Use the top term from the containment table, if one exists. 2. Concatenate the second top term from the table only if it has a significant score in this set (e.g., if a term appears only twice, we do not want to use it). 3. If the number of segments for this cluster is low (suggesting that the user is not very familiar with this area), or if the scores for the top two terms are low, concatenate the top term from the nearby cities table to the name. At the end of this process, we may have anything from 1 to 3 terms that are combined into a textual geographic description for this cluster. Incidentally, the names we used to represent the clusters in Figure 2 are in fact the top terms that were chosen for clusters of one of the test collections.

As for the inner nodes in the location hierarchy, some of them represent a country, and are easy to caption. For the other inner nodes, such as the "Around San Francisco" cluster in Figure 2, we pick the single top-scoring term from the containment table of each of the node's descendants, and take the top three terms in this list. The hope is that these three names represent the general area where the current cluster occurs; for a more accurate description of the area, one can turn to the lower level clusters. For example, our "Around San Francisco" cluster may be named "San Francisco, Berkeley, Sonoma". However, if this cluster is the only

[2]Regretfully, we only have access to a database of US cities and parks. Thus, we have only tested our naming procedure on US photos.

Figure 4: First-level clusters and their names for collection Z. Two clusters are not shown: "Seattle, WA" and "Philadelphia, PA".

one in this specific state, it will be assigned the state's name.

Finally, when assigning a textual geographic caption to an event, we assume that the user has visited relatively few places over the duration of the event. Furthermore, we assume the user will get more geographical context from the cluster information. For event names, then, we pick only the top term in the containment table. If one does not exist, we choose the top term from the nearby cities table. In any case, we can augment the name with date or time span of the event, e.g. "Boston, Dec 31st 2003".

5. EXPERIMENTS AND RESULTS

Our system produces three different types of output: event segmentation, location hierarchy, and suggested names. Evaluation of each of these outputs poses its own challenges. However, the main challenge in evaluating our algorithm is the current rarity of geo-referenced photo collections. We expect more collections to be available in the future, but today, other than the first author, we could only find two subjects with a large enough collection of such photos. Our results are then, by necessity, case studies rather than statistically significant analysis. Details of the two collections, R and K, together with the author's collection Z are listed in Table 1. For illustration purposes, Figure 4 shows the first-level clusters PhotoCompas created for the Z collection, and the proposed names. As the evaluation of Z may be biased by the author's knowledge, we are not showing results for it, but use its results to reaffirm the results from the other collections. As our processing is done separately for each country, we only used United States photos from these collection, since both subjects' collections had too few pictures from other countries to merit evaluation.

Table 1: Sample datasets used in our experiments

Collection	*Number of US Photos*	*Time Span*
R	2580	27 months
K	1192	14 months
Z	1823	13 months

5.1 Evaluation of Event Segmentation

In this section, we evaluate the success of our event segmentation. For this purpose, we asked the owners of our datasets for the "ground truth" segmentation of their collection, $M_{optimal}$, as it is defined in Section 3.2. As expected, $M_{optimal}$ demonstrated the difficulty inherent in the task of event segmentation and validated some of our guidelines. On one hand, the subjects often listed multiple picture-taking days as one event ("This was my trip to New York"). On the other hand, subjects often partitioned photos taken in a single day into multiple events: a birthday event closely followed by an unrelated dinner party, for example. Our evaluation goals were as follows:

1. Show that the event segmentation generated by our algorithm is accurate, and is an improvement over time-only techniques.
2. Explore the effect of system parameters on the event segmentation.
3. Verify that M_{over} is indeed an over-segmentation of the ground-truth events (see Section 3.2.1).

The metrics used in past studies of event clustering for digital photos (e.g., [10, 2]) are the precision and recall for the detected event boundaries:

$$precision = \frac{\text{correctly detected boundaries}}{\text{total number of detected boundaries}} \quad (1)$$

$$recall = \frac{\text{correctly detected boundaries}}{\text{total number of ground truth boundaries}} \quad (2)$$

These metrics are also used in other contexts such as video segmentation and natural language processing (NLP). While we feel these measure are relevant, we also feel they are lacking in capturing the complete semantics of event segmentation. For example, consider a collection of 10 photos and its ground truth segmentation $M_{optimal} = 1, 6, 10$. In other words, the collection has two event segments, photos 1-5 and 6-10. Now consider the suggested segmentations $1, 2, 5, 10$ and $1, 2, 9, 10$. While it is clear that the former is better than the latter, they are scored the same in both recall ($2/3 = .667$) and precision ($2/4 = .5$).

While designed for evaluating the NLP document-segmentation problem, we propose in addition to use two additional metrics to overcome the limitation of recall and precision. In [1], Beeferman et al suggest the P_k metric. This metric uses a sliding window to compute a score which is based on the probability that two photos (in our context) that are k photos apart are incorrectly identified as belonging to the same event, or not belonging to the same event. Pevzner and Hearst in [12] discuss shortfalls of the P_k metric, and suggest a variation named $WindowDiff$ (WD). The WD metric computes an error using a sliding window over the segmented set. At every position WD counts the number of segment boundaries that fall within the window. If the number is different between the ground-truth and the suggested segmentation, the algorithm assigns a penalty proportional to the difference. The authors suggest that the WD metric grows in roughly linear fashion with the difference between the compared segmentations. The possible values for P_k and WD range from 0 to 1; for both metrics, smaller values are better.

In Section 3.2.3 we presented the $h(c)$ function that comes into effect when consecutive events occur within the same

geographical cluster. This is the only hand tuned function that directly reflects on the event segmentation. Therefore, we had to verify the effect its parameters, H_{min} and H_{max}, have on the resulting segmentation. We omit this discussion for lack of space, but note that the algorithm seemed insensitive to values that ranged from 1 hour to 8 hours for H_{min}, and 100 to 200 hours for H_{max}. We also confirmed that the chosen $h(c)$ function performed a lot better than a simple policy of using a fixed time threshold (e.g., 1 hour, 24 hours) to detect consecutive events in the same geographical cluster.

We tested the performance of our event segmentation. The following conditions are compared:

- PC – Our PhotoCompas algorithm as proposed in Section 3. PC(8,192) and PC(1,192) represent sample pairs of $h(c)$ parameters.
- TB – Time-based segmentation algorithm from [7].
- FT – A "fixed" threshold segmentation: we detect a new event every time there is a gap of x hours.
- FS – Another "fixed" segmentation where we detect a new event if there is a gap of x hours or y kilometers.

The parameters for conditions TB, FT and FS were hand tuned to yield a minimal average $WindowDiff$ for the two collections. The values in brackets in the following figures are are the chosen parameters in hours (or hours and kilometers for FS).

The recall and precision metrics for the different conditions are presented in Figure 5. Before we discuss the results for the different strategies, let us check the recall and precision values for M_{over}, on the right hand side of the figure. Remember that, as presented in Section 3.2.1, M_{over} should be an over-segmentation of the collections (recall should be 100%). In reality, for both the R and K collections the recall is close to 85%, due to a total of 24 undetected events in both collections. When checking the undetected events, we discovered that all of them, except one, contained only a few photos (2.8 on average) and happened on the same day and in the same area as the adjacent event. The only longer missed event started literally minutes away from the end of another event. We conclude that this low recall value for M_{over} is inevitable, and suggest that these collections are inherently hard to segment.[3] While it is possible to tune the parameters in order to get a higher recall for M_{over}, this will cost significantly in the precision. Since M_{over} informs our location clustering, we must not be too aggressive in computing it.

Back to Figure 5, we observe that PhotoCompas does better in recall, precision, or both than all other algorithms with 80%-85% for both metrics and both collections.[4] We manually inspected the undetected (recall) and over-segmented (precision) events. Most of the undetected events, similarly

[3]The M_{over} recall for Z was 97%.

[4]The recall for Z was much higher, while the precision was slightly lower.

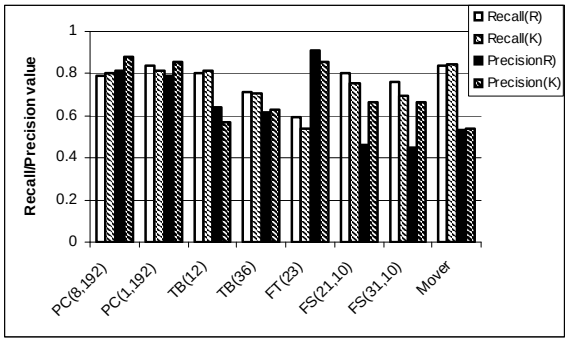

Figure 5: Recall and Precision values for different conditions. For both metrics, higher value means better performance.

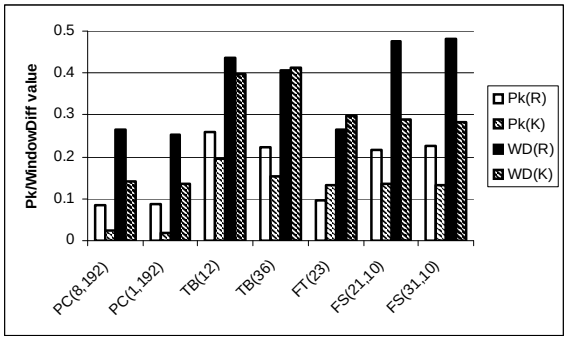

Figure 6: P_k and $WindowDiff$ values for different conditions. For both metrics, a smaller value means better performance.

to M_{over}, were minor events (few pictures) around the subjects' hometowns, within hours or less from other events. One notable exception is a road trip that our algorithm decided not to split, but which the subject actually preferred to split. On the other hand, when we checked the type of precision errors made by the algorithm, a lot of them involved road trips: a multi-day picture taking events, that span a very large geographical area. Our algorithm broke these trips apart when the time gap and distance between two consecutive pictures were very high (overnight, for example). Another type of over-segmented events were photos taken around the subjects hometown, when two consecutive photos were many hours apart but still "belonged together" as far as the subjects were concerned. Notice that automatically distinguishing between these and the undetected events mentioned earlier is impossible.

In Figure 6 we show the P_k and $WindowDiff$ metrics for the different conditions. We can see that PhotoCompas does better (lower values) than all the other algorithms. Also notice the these metrics bring out the differences in a much more apparent way than precision and recall; compare the results of strategy FT versus PC in both graphs. Results for collection Z were similar to the results for R.

5.2 Evaluation of Location Hierarchy

We qualitatively evaluated the location hierarchy created by PhotoCompas through interviews with the owners of the K and R collections. In both cases, many of the photos were centered around a single metropolitan area, but many others were taken during trips to various other places. Our goal was to check whether the clusters in the hierarchy:

- Are accepted by our subjects.
- Are similar to geographic grouping subjects would have generated themselves.
- Contain erroneous assignments (photos assigned to the wrong cluster).
- Are preferable to a "fixed hierarchy" of state/city.

The PhotoCompas clustering created 5 and 6 initial clusters for the two collections. In both, the algorithm split one of those clusters to lower level clusters: 5 in one case, and 10 in the other. We showed the subjects maps of the clusters, drawing the locations of all photos in each cluster. We solicited comments about the clusters and photos. We also asked the subjects to suggest edits to the clusters at each level: whether they would like to split or merge any of the displayed clusters.

The clustering results for the high-level clusters were accepted by the subjects, though both had minor edits for these clusters. One subject commented that he would have liked the cluster created for his Northern California / Southern Idaho photos be split into two clusters, by the state. At the same time, though, he liked the fact that photos from Nevada and Utah are grouped together, and that his Central California and Southern California photos were split into two clusters. The other subject did not approve of the algorithm's decision to keep all his East Coast photos in one cluster. Similar comments were made about the lower level clusters. In summary, the required edits to the clusters were limited. We believe that our first two evaluation goals are met quite successfully.

Checking the third goal, a small issue that arose for both subjects is the occasional overlap of two clusters. This happens, as noted in Section 3.2, since our algorithm prefers to keep "related photos" in the same cluster, even at the price of a slight geographic overlap. For example, a road trip that begins at home but the bulk of the photos are taken elsewhere. In such a case, all the trip photos will be assigned to one cluster, possibly creating an overlap with the "home" cluster. The overlap involves only a small number of photos – otherwise, our algorithm would merge the two clusters. Remember that our clustering has two goals: first, to inform the event segmentation; and second, to be used in the UI presentation to the user. It seems that the UI presentation may suffer slightly from this overlap. This question should be studied in a more extensive user study. Possibly, the clusters should be fixed *after* the event segmentation so photos are re-assigned with no overlap.

Finally, we showed the subjects the breakdown of their collection into strict administrative hierarchy of states and cities /parks. Clearly, in some cases this hierarchy was useful to the subjects (particularly for areas they know well), while in other cases it was confusing (photos that do not fall into any city/park, or when photo were taken in areas that the subjects are not familiar with). The tradeoffs in using the administrative hierarchy in the UI versus our automatic hierarchy needs to be investigated in a more extensive user study.

5.3 Evaluation of Naming

We again evaluated the PhotoCompas naming algorithm through interviews with the owners of the test collections. We concentrated on names for the geographical clusters and did not look at the geographic names we produced for events. Our evaluation goals were to verify that the produced cluster names are:

- Useful to the subjects, in that a) the name includes terms that are familiar to the subjects and help them understand which geographic area is covered by the cluster, b) the subject is able to tell the cluster apart from other clusters, based on the name and c) the subject can tell which pictures belong to this cluster based on the name.
- Similar to the names that the subjects would have generated themselves.

For each collection, and each cluster, we performed two tests. In the first test, we showed the subjects the list of terms that appeared in the *contained table* and *nearby cities table* for each cluster as defined in Section 4.1. The average length of the lists for the different clusters was 19 place names; it was generally shorter for leaf clusters. For each cluster, we asked the subject to pick at most three place names that represent this cluster best (in fact, they only picked 1.84 place names on average; our algorithm used an average of 2.3 place names for each cluster). As an aid, we showed the subjects maps for all the clusters and offered to show them the pictures as well – that was usually not necessary as subjects had a very good idea what those pictures were. For 76% of the clusters, our algorithm and the subjects picked at least one place name in common. Furthermore, our next test shows that for most of the other 24% of the clusters, the subjects found the given name useful.

In the second test, we asked the subjects to comment on the usefulness of each cluster name. We found that the automatically-produced names were extremely useful (as defined above). Out of the total of 25 clusters, subjects were content with all but one name. In this one case, our clustering algorithm grouped together all photos from three different US East Coast cities; but the name only represented one of them. Other comments were: a) Three of the cluster names included one park or city name that was obscure to the subject; b) one cluster name was not representative enough of a small subset of its photos. In general, both subjects expressed strong satisfaction with the usefulness of the names.

6. CONCLUSIONS AND FUTURE WORK

We have shown that PhotoCompas can automatically generate a meaningful organization for personal photo collections. In particular, the system performed well when detecting events in collections; generated location hierarchies that were intuitive to collection owners; and assigned node names that proved useful.

We have built a prototype interface that will support PhotoCompas for desktop and PDA environments. This interface is generated using the HTML-based Flamenco metadata search interface by Yee et al [19]. In our future work, we plan to use this interface to examine the UI presentation aspects of PhotoCompas. For example, is a UI based on our generated location hierarchy more effective than one based on a pre-defined state/city location hierarchy? Also, what are the tradeoffs between our approach and a map-based interface approach for geo-referenced photos?

One of the key problems is the lack of multi-year geo-referenced photo collections. Once we have obtained more collections, possibly via manual georeferencing of existing photo collections, we plan to verify that our techniques are also effective for collections that span 20-30 years of photos. In addition, we would like to compare our approach to alternative implementations.

Finally, a more general problem may arise from our naming algorithm. Occasionally, when naming clusters, subjects suggested a cluster name that would be difficult to find on the basis of our data, e.g. "Northern California" or "East Coast". While it seems feasible to generate the former automatically, the second is much harder. We will try to address this challenge in our future work.

7. ACKNOWLEDGMENTS

We would like to thank Meredith Williams for guidance around the complicated world of Geographical Information Systems, and Kevin Li for patient support and guidance with the Flamenco installation. We also thank Kentaro Toyama and Ron Logan of the Microsoft Research WWMX team for granting us access to their data and volunteering for the experiment. As for our own, we thank Jichun Zhu for his devoted programming work and Susumu Harada for critical help when it was needed.

8. REFERENCES

[1] D. Beeferman, A. Berger, and J. D. Lafferty. Statistical models for text segmentation. *Machine Learning*, 34(1-3):177–210, 1999.

[2] M. Cooper, J. Foote, A. Girgensohn, and L. Wilcox. Temporal event clustering for digital photo collections. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 364–373. ACM Press, 2003.

[3] D. Frohlich, A. Kuchinsky, C. Pering, A. Don, and S. Ariss. Requirements for photoware. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, 2002.

[4] U. Gargi. Consumer media capture: Time-based analysis and event clustering. Technical Report HPL-2003-165, HP Laboratories, August 2003.

[5] A. Gionis and H. Mannila. Finding recurrent sources in sequences. In *Proceedings of the seventh annual international conference on Computational molecular biology*, pages 123–130. ACM Press, 2003.

[6] Google inc. http://www.google.com.

[7] A. Graham, H. Garcia-Molina, A. Paepcke, and T. Winograd. Time as essence for photo browsing through personal digital libraries. In *Proceedings of the Second ACM/IEEE-CS Joint Conference on Digital Libraries*, 2002. Available at http://dbpubs.stanford.edu/pub/2002-4.

[8] S. Harada, M. Naaman, Y. J. Song, Q. Wang, and A. Paepcke. Lost in memories: Interacting with large photo collections on pdas. Technical report, Stanford University, September 2003. Available at http://dbpubs.stanford.edu/pub/2003-30.

[9] C. B. Jones, R. Purves, A. Ruas, M. Sanderson, M. Sester, M. van Kreveld, and R. Weibel. Spatial information retrieval and geographical ontologies an overview of the spirit project. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 387–388. ACM Press, 2002.

[10] A. Loui and A. E. Savakis. Automatic image event segmentation and quality screening for albuming applications. In *IEEE International Conference on Multimedia and Expo*, 2000.

[11] M. Naaman, A. Paepcke, and H. Garcia-Molina. From where to what: Metadata sharing for digital photographs with geographic coordinates. In *10th International Conference on Cooperative Information Systems (CoopIS)*, 2003.

[12] L. Pevzner and M. Hearst. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36, 2002.

[13] J. C. Platt, M. Czerwinski, and B. A. Field. Phototoc: Automatic clustering for browsing personal photographs. Technical Report MSR-TR-2002-17, Microsoft Research, February 2002.

[14] K. Rodden and K. R. Wood. How do people manage their digital photographs? In *Proceedings of the conference on Human factors in computing systems*, pages 409–416. ACM Press, 2003.

[15] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistic*, 6:461–464, 1978.

[16] D. Spinellis. Position-annotated photographs: A geotemporal web. *IEEE Pervasive Computing*, 2(2):72–79, 2003.

[17] K. Toyama, R. Logan, and A. Roseway. Geographic location tags on digital images. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 156–166. ACM Press, 2003.

[18] W. Wagenaar. My memory: A study of autobiographical memory over six years. *Cognitive psychology*, 18:225–252, 1986.

[19] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *Proceedings of the conference on Human factors in computing systems*, pages 401–408. ACM Press, 2003.