

Exploiting Location and Time for Photo Search and Storytelling in MyLifeBits

Aleks Aris
Jim Gemmell
Roger Lueder

Microsoft Research

September 2004

Technical Report
MSR-TR-2004-102

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Exploiting Location and Time for Photo Search and Storytelling in MyLifeBits

Aleks Aris, Jim Gemmell and Roger Lueder
Microsoft Research, San Francisco, CA, USA

Abstract

Location information can be used as an additional metadata to time in order to organize photos. Moreover, we believe location information derived from GPS tracks may be useful to look at and visualize for certain tasks. In this paper, we explore ways to organize images and location information, visualize them, and provide interactive operations such as filtering and replaying. Filtering over long ranges and visualizing while filtering as well as what is being filtered were problems we encountered. Our solution to these was a widget we developed called DoubleAlphaSlider, which is capable of filtering over long ranges, supports zoom together with an overview widget that displays the representation of the data and recent contexts that were zoomed from. In addition, we focused on how to help the user understand several derived properties of the data such as direction, adjacency in terms of time and place of subgroups of the data, which are called “trips” in our case.

Introduction

Location and time are powerful keys to unlocking memories. We believe that location awareness will become ubiquitous in media capture devices such as cameras, cell-phone cameras, and video cameras. As these devices already include clocks, it means we will be able to search for media based on place and time. In addition to search, location and time aid in storytelling. A list of locations covered in time order is a simple telling of the story of the trip, and by combining media with maps and animations we can create visualizations to tell the story in a much more compelling fashion.

This paper describes a prototype to support time/location-based media search and trip visualization. The prototype is built on top of the MyLifeBits platform. MyLifeBits is designed to be a personal lifetime store for all digital media and data including email, calendar events, contacts, documents, audio and video. Entities in MyLifeBits have type-specific attributes. For instance, a photo has a date-taken and a location-taken (latitude/longitude coordinates), while email has a sender attribute. Entities in the MyLifeBits store may also have typed links between them. For instance, there may be a link of type “person in photo” between a contact and a photo. Or, there may be an “annotation” link from a text document to a video, indicating that the text comments on the video. MyLifeBits uses a SQL Server database to store metadata. Careful

database design allows MyLifeBits to support fast search, clustering, and lookup by common attributes.

The technology used to obtain location may be GPS, triangulation of cellular telephone signals, detection of RFID tags at some known location, or one of many other methods. Regardless of which approaches become most widespread, we can now anticipate ubiquitous location-enabled devices and begin work on the software that will let us leverage them. At present, we carry a pocket-size GPS independent of our media capture devices. Time correlation between the GPS record and the media is then used to infer a location for the media. GPS records are also interesting apart from any media; they show the “trail” you have traveled. To date, we have only experimented with photos, although much of our discussion would extend to video or audio in a straightforward way. In addition to off-the-shelf cameras, we have also experimented with SenseCams (described below).

In the remainder of this paper, we first explain how location information and photos are loaded into MyLifeBits, and the core support for search and browsing based on location and time. Then, we describe our trip visualization tool that animates a trip to help tell its story. Finally, we present a novel DoubleAlphaSlider control used to select time ranges.

Loading, Search and Browsing

GPS location readings, consisting of date/time, latitude, longitude, elevation, and precision, are loaded into a table in the MyLifeBits database by an import program. Date/time correlation between photos and GPS readings may then be used to set the location in the photos whenever possible. However, the user may elect to not set the location in photos based on time, because they may obtain some photos from a third party, or may have loaned the GPS to someone else. Furthermore, some photos may already have their location set by a third party. So, while inferring location based on time correlation with a GPS is a feature that we support, it cannot be used in all instances, and location must be stored as an attribute for each photo. Note that many GPS “track points” will not have any corresponding photo, but are still very interesting and worth saving as a record of the user’s location.

Photos have their meta-data stored in the MyLifeBits database, while their image content is a file in an NTFS folder tree that MyLifeBits is configured to monitor. In the

case of commercial cameras, it is simple enough to load photos into an NTFS folder. A little more is involved in the case of a SenseCam. SenseCam is a device that combines a camera with a number of sensors, including a 2-channel accelerometer, a digital light sensor, a temperature sensor and a passive infra red sensor for detecting living beings. Photos are triggered by sensor data and/or time interval. It is worn on a pendant hung from one's neck. The sensors are used to automatically take pictures at "good" times [17]. Sensor information is also recorded, and is uploaded along with the photos. An import program uploads SenseCam photos and sensor data into MyLifeBits. The photos are in JPEG format and are stored just like any other JPEG photos in MyLifeBits, with attributes that include date/time taken, location, and camera make. The sensor data is stored in tables in the MyLifeBits database. All sensor values include the date/time of the sensor reading. With SenseCam, photo-taking is passive, freeing the user up to just experience rather than worry about taking photos [17].

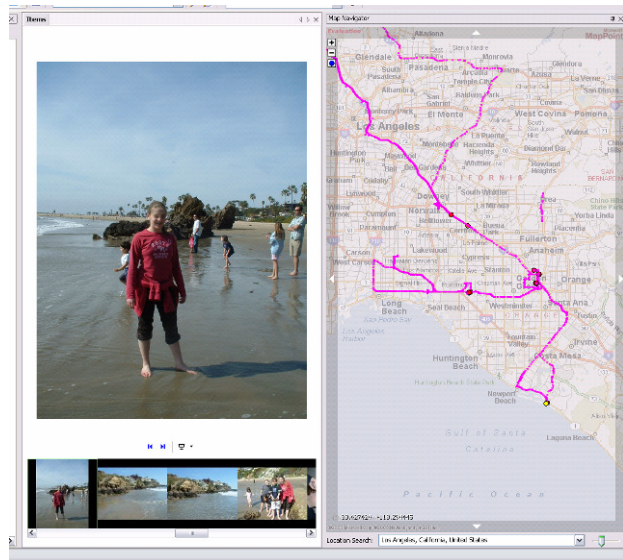


Figure 1 – Map UI map on right shows large red dots where photos are taken and small pink dots for GPS track points. The pane on left shows photos taken in the area shown on the map, in film strip view with a large preview.

A map UI element in the MyLifeBits interface marks photo locations by a large red dot and track point locations with a smaller purple dot (Figure 1, Figure 2). Moving the mouse over the corresponding dot for a photo pops up a thumbnail of the image in a direction that avoids being partially drawn (Figure 3). A location (address, zip code, or city/state) may be entered into the "Location Search" text box to set the map location. The +, -, o on the left of the map (Figure 4) support zoom in, zoom out, and zoom out to the entire earth, respectively. A region of the map may also be selected to zoom into using the mouse (Figure 4, Figure 5), while a clickable arrow appears on mouse over near the map edges to support panning. Zooming and panning on the map issues a new

query for photos from the region of the visible map to be shown in the corresponding pane. Photos that have no location (e.g., old photos) can be dragged and dropped onto the map to set their location. The basic mapping interface was ported from the WWMX, and more details on its appearance and operation can be found in a paper describing WWMX [18].

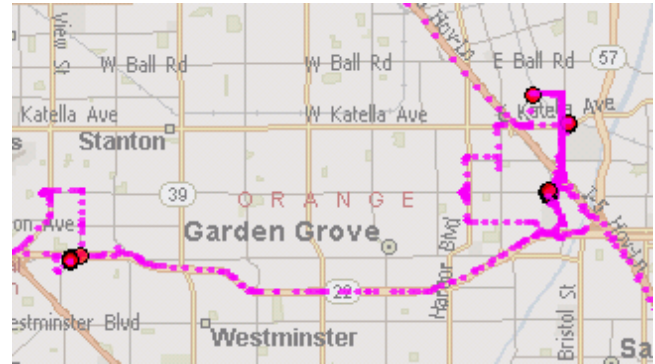


Figure 2 – Close-up of map from Figure 1.

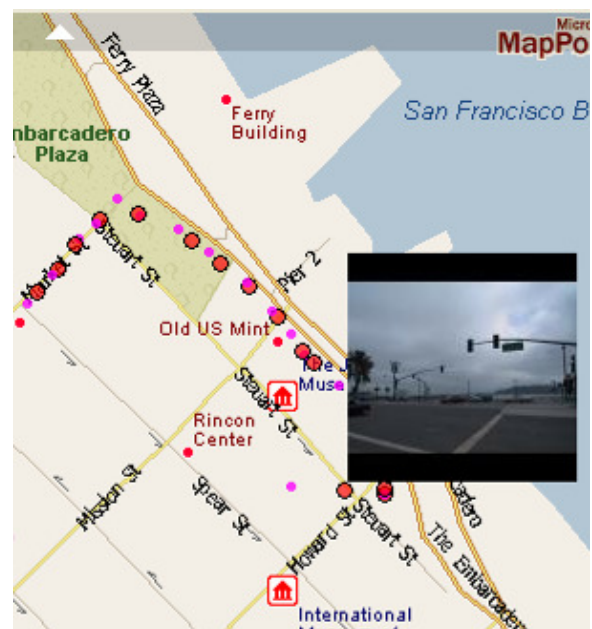


Figure 3 pop-up image when mouse hovers over the corresponding dot on the map

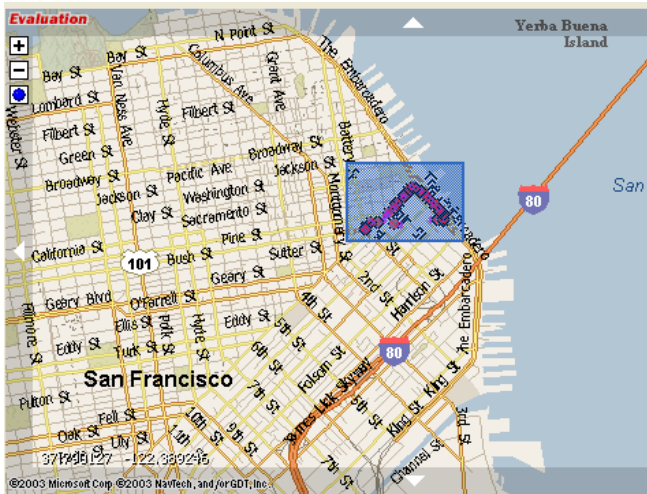


Figure 4 Selecting a map region to zoom

After some time of using a GPS and taking photos, a selected map region may become saturated with points. One way to filter out undesired clutter on the map is to restrict photos and track points by time. To this end, we divide the photos and track points into clusters and consider each a “trip.” The clustering algorithm simply looks for gaps in space or time above a given threshold (e.g. time gap > 90 min, or location gap > 1 mile) in order to divide the data. The trips are displayed in a list box below the map. The top line allows the selection of all track points and photos by clicking on “All Trips.” The number in parentheses to the left of each line, e.g., (1154) to the left of “All Trips”, indicates how many photos/points that trip includes. When a trip is selected from the list, only its photos and track points appear on the map (Figure 6). An alternative to filtering by trip is to view all trips, but connect the dots belonging to the same trip with a line to help distinguish between trips. The user can select this with the “show connections” checkbox (Figure 7).

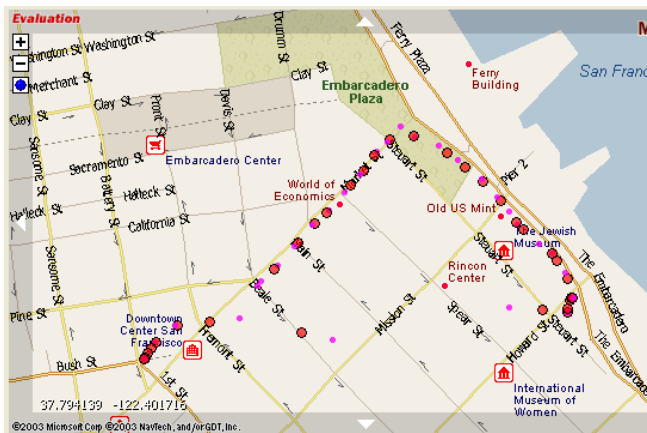


Figure 5 Zoomed map (after selection from Figure 4)

The use of time for search and browsing is pervasive in MyLifeBits. Time attributes can be used in search and refinement of searches. For example, the user may search for all files created in a given date range. Or, suppose a search for emails results in a large number of hits. The user

may select to refine by date received, and MyLifeBits will cluster the emails by date on the fly. The user can then pick a cluster to narrow in on. MyLifeBits logs most user activity and it is a simple right-click operation to see the history of that item. For example, the history of a photo could show when it was taken, and every time the JPEG file has been opened. MyLifeBits also supports “pivots” on time corresponding to the user remembering that something happened at the same time. For instance, a user may click on an old appointment in their calendar and ask to see everything that overlaps in time with it, and thus find photos taken at that time. For more discussion on using time in MyLifeBits see [17], [15].

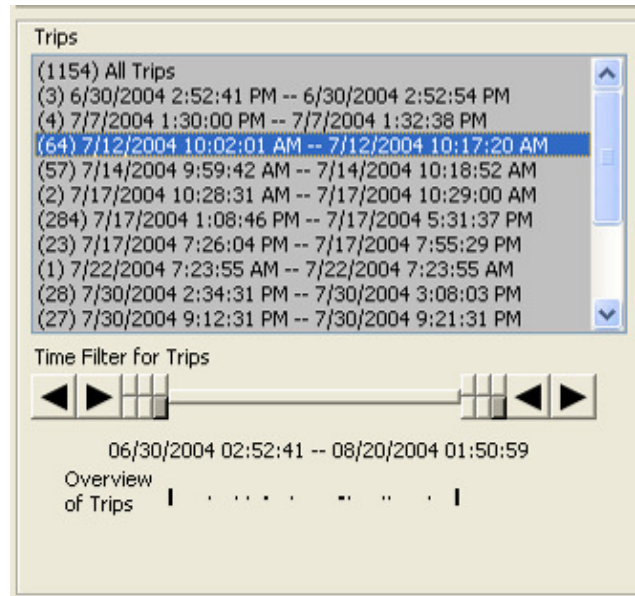


Figure 6 Selecting a trip

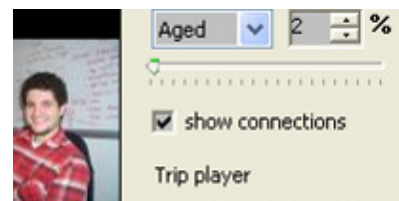


Figure 7 show connections checked

Trip Replay Visualization:

The interface described so far supports searching and browsing based on location and time. However, a static map of track points and photo locations is still hiding a lot of useful information. One does not know the direction traveled, or the time spent in different places. We have created a trip “replay” visualization that animates the trip to show this information to the user. During the replay, a curve grows from the starting point of the trip towards the ending point, with the time spent at each point proportional to real time of the trip. As the curve hits locations where an image has been taken, a preview of the image is displayed below the map (Figure 8). During the replay, the images are retrieved from the database on demand and cached to

improve the performance. The same cache is used to store and retrieve the pop-up images (as seen on Figure 3).

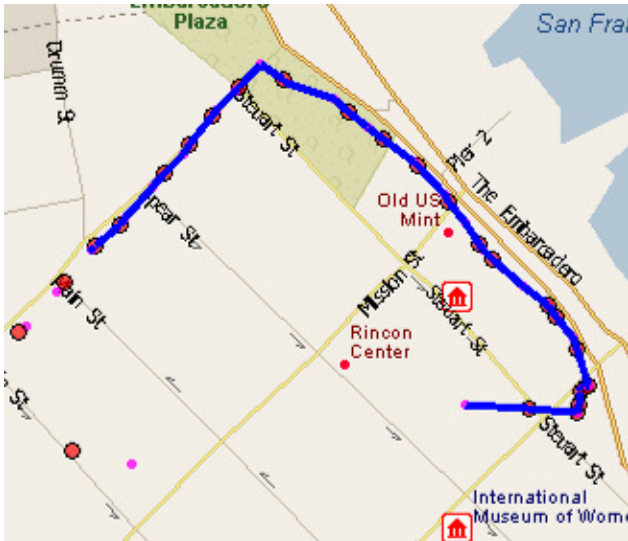


Figure 8 Replay of a trip

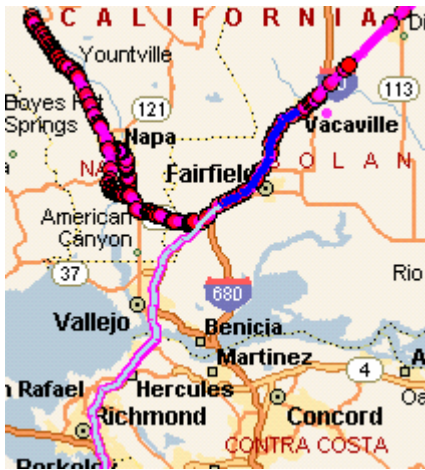


Figure 9 Replay with aging of colors

To start the replay one presses the play button (Figure 10, under “Trip Player”, the leftmost button, which is a triangle). During the replay, the “Replay Progress” slider shows the progress of the replay (in terms of time) advancing from left to right (see Figure 10). The user can

press pause button to suspend the replay. While paused, the progress slider may be dragged to a desired point of the replay, and replay can be resumed from that point. During playback, one can also directly drag the progress slider to any desired point of the replay and the replay will continue from that point when the user releases the slider. While the user drags the slider, the curve dynamically grows and shrinks. Simultaneously, the picture box updates to show the current image. In this respect, the slider provides a form of dynamic query [3], [4], [8], [9] and utilizes the principles of direct manipulation [7], [10], [13].

The user can select the duration for the trip replay, such as 5 seconds (Figure 10, bottom right). In trips with highly variable speed (e.g. walking down the street taking a few pictures, stopping for a minute to take several photos of an interesting feature, and then continuing to walk with a few snap shots) some photos may be displayed for a long time, while others will flip by so fast as to be unrecognizable. To compensate, the user may switch replay from “proportional” to “sequential”, which cause the replay to show each photo for the same amount of time and disregard the actual rate of the trip. Figure 10, bottom right, shows the UI for selecting sequential display.

If a trip overlaps itself (e.g., in a round trip, where one returns by the same path) the animation will not be visible where overlap occurs. In order to avoid this problem, we use a different color for the leading part of the animated line from that of the previously rendered line. Thus, the leading portion of the animation is always visible, even when it overlaps previous sections of the trip. We are experimenting with how much of the line to have in a different color and currently expose the setting in the UI. “Full” selects just a single color, while “Aged” allows the user to select the amount in a different color as a percentage of total trip time. Using a percentage of total trip time makes the amount in a different color vary according to the speed traveled, giving the user a way to compare speed during the trip. Figure 9 shows a trip with color aging; new points are plotted in dark blue and then aged to light blue.

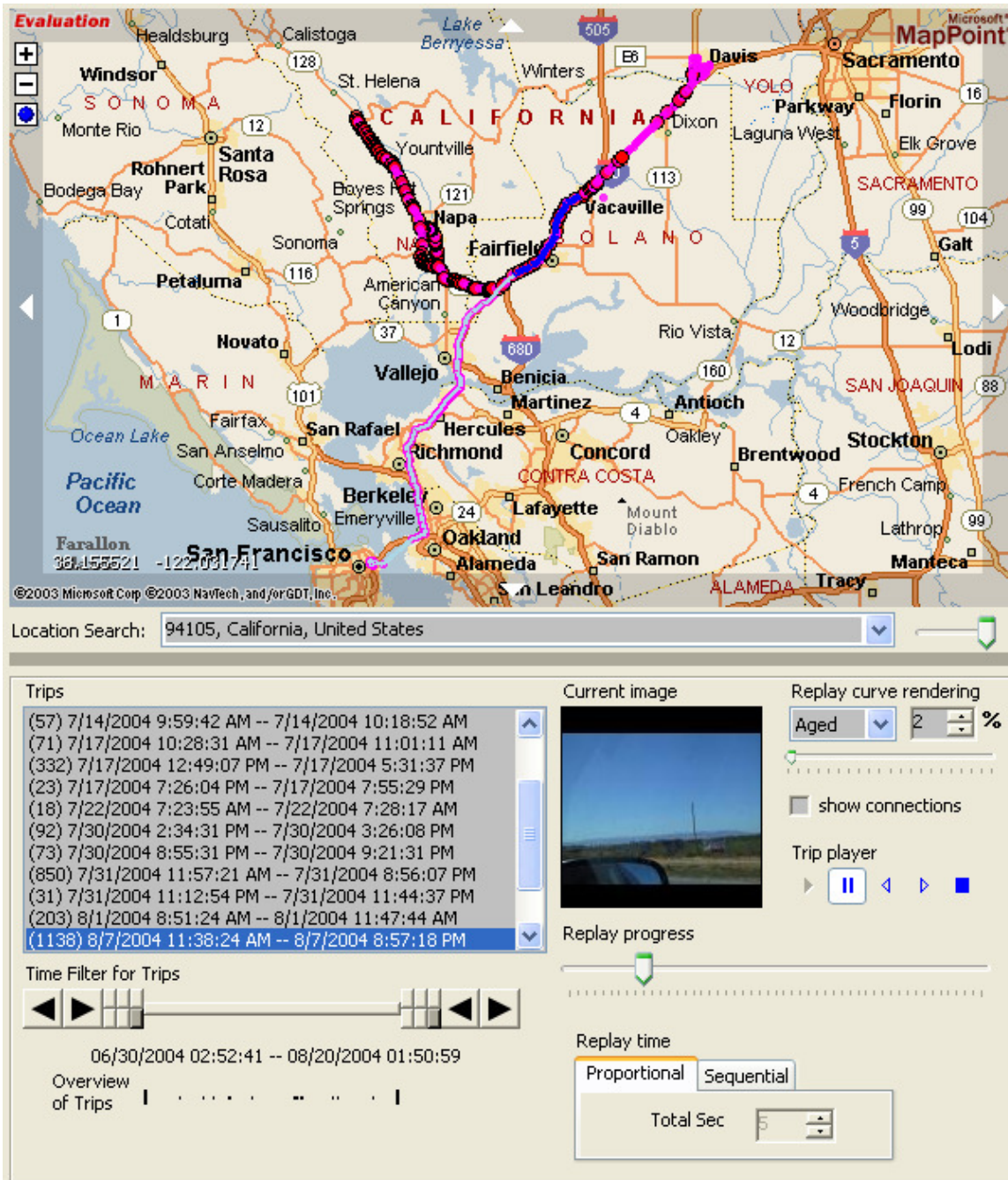


Figure 10 Full trip replay UI

Selecting a Time Range

MyLifeBits is intended to be a lifetime store. Therefore, we must anticipate users with decades worth of data. Consequently, a person will have an inordinate number of trips at certain locations, making the trip list too long to be useful. In such a case, restricting the time range under consideration even before listing trips can be very useful. In this section, we discuss some work performed on the use of sliders for time range selection.

DoubleSliders (also known as RangeSliders) are widgets known to be effective for range queries [3], [8]. They are used and illustrated in applications such as FilmFinder [4], [5]. However, if one desires to cover a range of, say, 30 years, there are not enough pixels on screen to allow fine grain selection down to days (over 10,000 to choose from) much less seconds (9.5×10^8).

AlphaSliders are designed to overcome the problem of long ranges [2], [6]. They are effective to select a value over a long range. Another widget that allows specifying

a value over a long range is the TimeSlider [19]. TimeSlider, as it is with the AlphaSlider, doesn't support range selections. Specifically, an AlphaSlider is not a DoubleSlider as it only has one slider. We combined the powers of the two widgets, which lead to "DoubleAlphaSlider". One version of AlphaSlider has 3 speed levels that enable the user to traverse slowly, medium-level, or fast [2]. The trick of the AlphaSlider is to break the 1-1 correspondence of a pixel to a value. A pixel may represent more than one (adjacent) value in AlphaSlider.

Time naturally breaks down to 6 different units, that is, year, month, day, hour, minute, and second. We need several speed levels for such a long period as in the speeded version of the AlphaSlider. To provide the user with a direct mapping [1], we chose to have 6 levels of speed. As a result, we designed a 6-speed DoubleAlphaSlider. We put this widget just below the trips list to denote that using this widget one filters the trips in the list, as well as the trips shown on the map when "All Trips" is selected (Figure 10). The speed can be selected by clicking to one of the 6-buttons on either side. The top row, from left to right sets the speed for year, month, and day, respectively. The bottom row, from left to right sets the speed for hour, minute, and second, respectively.

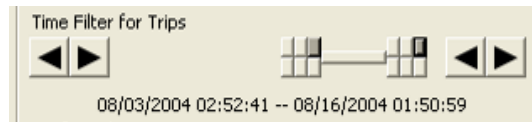


Figure 11 using the 6-speed DoubleAlphaSlider to filter trips by time

The sliders on the left and on the right (each slider consists of 6 speed buttons) can be dragged independently. The selected speed button is darkened to provide feedback. Selecting a speed level on the left automatically selects the same speed level on the right. Dragging the slider is slower when the speed level is low (e.g. seconds), and is faster when the speed level is high (e.g. years). If the speed *day* corresponds to the normal mouse movement, setting the speed to *hour*, one will have to move the mouse 24 times faster to the same direction to achieve the same effect. Figure 11 shows a close-up of the DoubleAlphaSlider, with the speed set to day. By clicking and dragging the left *day* button and then the right *day* button, the range has been restricted.

One can do incremental adjustments by using the arrowhead buttons of the 6-speed DoubleAlphaSlider on the left and on the right for fine-tuning. The bar between the button clusters represents the range and is draggable. When dragged, the complete range moves to the direction of dragging (left or right) with the speed set (the range bar in Figure 11 is dragged to attain the situation in Figure 12).

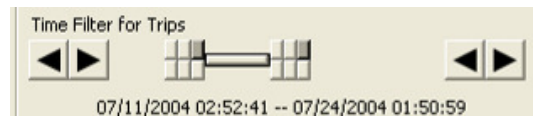


Figure 12 after dragging the range to the left

Double-clicking to the range bar of the 6-speed DoubleAlphaSlider with the left mouse button zooms into the selected time period. Double clicking the range bar using the right mouse button zooms back out to the previous range.

When a slider is dragged, the updating is done dynamically so the user can see the effects of the dragging immediately. Trips on the list change dynamically and so do the dots on the map as the user drags a slider (left, middle, or right) forth and back. In other words, there is tight coupling [4] between the dots on the map and the selected item on the trips list, as well as the trips in the list and the selected time range on the slider. These properties of the slider make it a dynamic query widget [3], [4], [9].

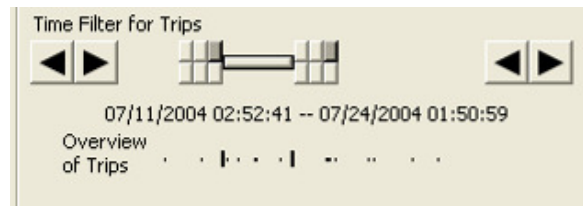


Figure 13 The overview at bottom shows the distribution of trips in time.

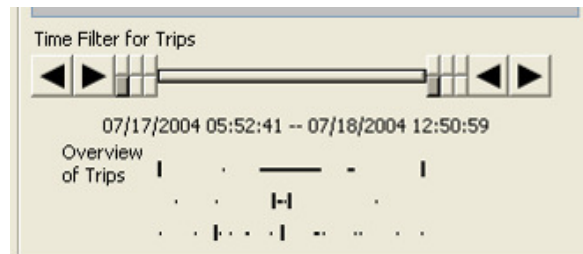


Figure 14 After zooming in twice, there are three lines to the overview, showing the distribution at each zoom level.

We have also begun to experiment with an "overview" that shows a compact representation of the distribution of trips over time. Figure 13 shows the overview underneath the slider. The vertical bars on the overview correspond to the time range selected. The vertical bars of the topmost overview are in sync with the sliders of the DoubleAlphaSlider. Each time a zoom-in is performed, the previous overview is kept, but moved lower in the display, in order to provide orientation. Figure 14 shows an overview including three different zoom levels: the current one (on top) and the previous two. In the design of the overview, we were careful not to use any unhelpful rendering keeping the data-ink ratio high [11], [12]. The trips and the gaps between them are represented by the rendered and non-rendered dots, respectively, exploiting the principle of 1+1 = 3 or more [12] for our purposes. Its

size is kept compact, but providing zoom-in when needed, generating small multiples on demand [11], [12]. By saving the states, and providing the zoom feature, the task of fine-tuning the range according to the visual representation of the data is made possible.

Conclusion

Ubiquitous support for location awareness in media capture devices is coming soon, and we are preparing the software to leverage that information now. This paper describes our first attempts at using location, in conjunction with time, to browse, search, and tell stories. While our experience so far is positive, there is much work still to be done. A number of UI elements that are necessary to experiment should probably be removed and replaced with automatic settings (e.g., the percent of a trail leave in the non-aged color). User studies are needed to find out which settings can be automated, and to determine if the DoubleAlphaSlider will be deemed useful. It is hard at present to watch a trail animate and to observe the photo previews at the same time. Perhaps new animation methods will make observing both at once possible.

Acknowledgements

Gordon Bell has been a key collaborator on the MyLifeBits effort. Lyndsay Williams and Ken Wood collaborated on MyLifeBits support for SenseCam. Ron Logan and Kentaro Toyama contributed the WWMX mapping software for MyLifeBits. Jim Gray has provided many helpful suggestions. Ben Shneiderman was kind to provide comments and share his impressions with us.

References

- [1] Norman, Donald A., *The Design of Everyday Things*, 2002, Basic Books, 23-26.
- [2] Ahlberg, C., Shneiderman, B., The AlphaSlider: A Compact and Rapid Selector. *Proceedings ACM CHI'94: Human Factors in Comp. Systems*, 365-371.
- [3] Ahlberg, C., Williamson, C., Shneiderman, B., Dynamic Queries for Information Exploration: An Implementation and Evaluation, *Proceedings ACM CHI'92*
- [4] Ahlberg, C., Shneiderman, B., Visual information seeking: tight coupling of dynamic query filters with starfield displays, *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence*, 1994, 313-317.
- [5] Ahlberg, C., Shneiderman, B., Visual information seeking using the FilmFinder, *Conference companion on Human Factors in computing systems*, 1994, 433-434.
- [6] Osada, M., Liao, H., Shneiderman, B., AlphaSlider: searching textual lists with sliders, *Proc. Of the Ninth Annual Japanese Conf. on Human Interface*, Oct.1993.
- [7] Shneiderman, B., Direct Manipulation: A step beyond programming languages, *IEEE Computer 16*, 8 (August 1983), 57-69.
- [8] Williamson, C., Shneiderman, B., The Dynamic HomeFinder: Evaluating dynamic queries in a real-estate information exploration system, *Proc. ACM SIGIR Conference*, 1992, 339-346.
- [9] Shneiderman, B., Dynamic queries for visual information seeking, *IEEE Software 11*, 6 (June 1994), 70-77.
- [10] Shneiderman, B., Direct manipulation for comprehensible, predictable and controllable user interfaces, *Proceedings of the 2nd international conference on intelligent user interfaces*, 1997, 33-39.
- [11] Tufte, E. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire CT, 2001.
- [12] Tufte, E. *Envisioning Information*. Graphics Press, Cheshire CT, 2001.
- [13] Shneiderman, B., Plaisant, C., *Designing The User Interface*, 4th ed., Addison Wesley, 2004.
- [14] Gemmell, Jim, Bell, Gordon, Lueder, Roger, Drucker, Steven, and Wong, Curtis, MyLifeBits: Fulfilling the Memex Vision, *Proceedings of ACM Multimedia '02*, December 1-6, 2002, Juan-les-Pins, France, pp. 235-238.
- [15] Gemmell, Jim, Lueder, Roger, and Bell, Gordon, Living With a Lifetime Store, *ATR Workshop on Ubiquitous Experience Media*, Sept. 9-10, 2003, Keihanna Science City, Kyoto, Japan, pp.69-76.
- [16] Gemmell, Jim, Lueder, Roger, and Bell, Gordon, The MyLifeBits Lifetime Store (demo description), *ACM SIGMM 2003 Workshop on Experiential Telepresence (ETP 2003)*, November 7, 2003, Berkeley, CA.
- [17] Gemmell, Jim, Williams, Lyndsay, Wood, Ken, Lueder, Roger, and Bell, Gordon, Passive Capture and Ensuing Issues for a Personal Lifetime Store, First ACM Workshop on Continuous Archival and Retrieval of Personal Experiences (CARPE04).
- [18] Toyama, Kentaro, Logan, Ron, Roseway, Asta, and Anandan, P., Geographic location tags on digital images, *Proceedings of ACM Multimedia '03*, November 2003, Berkeley, CA, USA, pp. 156-166.
- [19] Koike, Y., Sugiura, A, Koseki, Y., TimeSlider: An interface to specify time point, *Proceedings of User Interface and Software Technology (UIST 97)*, ACM Press, 43-44.