TraceViz: "Brushing" for Location Based Services

Yung-Ju Chang, Pei-Yao Hung, and Mark W. Newman
School of Information, University of Michigan
Ann Arbor, 48109
{yuchang,peiyaoh,mwnewman}@umich.edu

ABSTRACT
The popularization of Location Based Services (LBS) has created new challenges for interaction designers in validating the design of their applications. Existing tools designed to play back GPS location traces data streams have shown potential for testing LBS applications and for supporting rapid and reflective prototyping. However, selecting a useful set of location traces from among a large collection remains a difficult task. In this paper, we present TraceViz, the first system that is aimed specifically at supporting LBS designers in exploring, filtering, and selecting location traces. TraceViz employs dynamic queries and "brushing" to allow LBS designers to flexibly adjust their trajectory filter criteria to find location traces of interest. An evaluation performed with eight LBS designers and developers indicates that TraceViz is helpful for rapidly locating useful traces and also highlights areas for future improvement.

Author Keywords
Location-based services; design tools; information visualization; direct manipulation; capture and playback

ACM Classification Keywords
D.2.2 [Design Tools and Techniques]: User interfaces; H.5.2 [User Interfaces]: Interaction styles

INTRODUCTION

Commercial location-based services (LBS) have been popular on many mobile platforms in recent years because of the low cost of sensors and widespread high-speed mobile Internet access. In addition to these commercial LBSes, many researchers have sought to develop a variety of LBSes for mobile guides, transport support, mobile gaming, and assistive technology and health [7]. However, to develop a high-quality LBS, testing and evaluation are essential tasks during development. Because an LBS is meant to provide information based on location context—i.e., a user's current location, trajectory, speed, or direction—it is critical that the designers examine and evaluate their LBS's design and behavior in realistic situations by testing with real location data. While field testing is an important technique for evaluating application behavior under realistic conditions, it can be expensive, time consuming, and infeasible early in the design process. An alternative approach is to *bring the field into the lab* by capturing naturalistic location data to "play back" during development time to enable rapid iterative design and testing [6].

Several widely used LBS development tools such as the Dalvik Debug Monitor Server (DDMS)[1] for Android and the location simulator of Xcode[2] for iOS development both provide a location testing feature that allows LBS designers to examine how application prototypes behave with actual GPS location data traces. Thanks to the prevalence of mobile GPS recording applications, it has become increasingly easy and economical for designers to capture a large collection of location traces. However, as an increasing amount of GPS location data has been captured and aggregated, it becomes difficult to select particular location traces for testing. While general-purpose geovisualization tools like Google Earth Desktop[3] can be used to visualize location traces on a map, it remains challenging to explore, filter, and select individual location traces when presented with a large set of data.

Consequently, a tool that can effectively filter and highlight relevant location traces is needed. To address these needs, we developed a trace filter and selection tool called TraceViz to allow LBS designers to filter by *brushing* to directly indicate geographical regions and trajectories of interest. TraceViz provides three brush modes—*Reselect*, *Intersect*, and *Union*—to allow designers to flexibly narrow down or expand filter criteria. When a brush stroke is drawn, the system calculates the similarity score of nearby traces and adjusts their visual saliency: only highly similar traces remain salient on the map to make it easier to highlight and select those relevant traces. After selecting traces of interest, the designer can import the selected traces from into their chosen playback tool.

MobileHCI'12, September 21–24, 2012, San Francisco, CA, USA.

[1] http://developer.android.com/guide/developing/debugging/ddms.html

[2] http://developer.apple.com/xcode/

[3] http://www.google.com/earth/explore/products/desktop.html

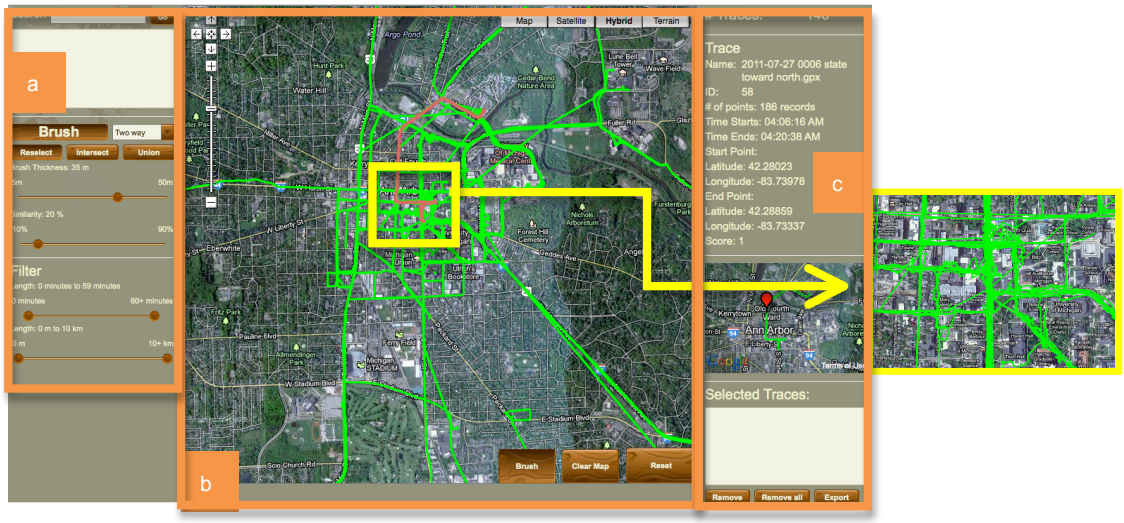

Figure 1. (Left) The main interface of TraceViz consists of three components. The Control Panel (a) contains a search box for geographical queries, brush related controls, and generic filtering controls. The TraceViewer (b) allows users to view traces and filter them by brushing. The Trace Info Panel (c) displays detailed information about a currently highlighted trace. (Right) With a large number of overlapping traces it is difficult it is to distinguish one location trace from others.

TraceViz is the first tool to leverage *brushing* to help LBS designers explore, filter, and select location traces for testing LBSes. By making it easier to find and select relevant traces, TraceViz can potentially encourage LBS designers and researchers to validate the design of their location-aware applications with a greater variety of location traces, in turn producing higher-quality systems. In the remainder of this paper we discuss related work, describe the TraceViz system with a focus on the implementation of brush-based filtering, and present the results of a preliminary evaluation.

RELATED WORK

A number of geovisualization systems have been developed for supporting movement analysis, identifying trajectory patterns and path summarization (e.g.[1,3]). These systems, however, have not focused on supporting the selection of specific examples for application testing. Brushing has been proposed in GIS systems for data linking and highlighting (e.g. [5]), as well as for specifying queries. de Silva and Aizawa [2], implemented a system for users to sketch a path or an area on a map to query linked multimedia data related to that path or area. FromDady [4] employs brushing to support iterative queries for helping visual analysis of aircraft trajectories. TraceViz is unique from these prior systems in that: a) brushing is used as a way for specifying *trajectory as a filter criterion* rather than simply for querying data in a brushed area, and b) it is the first work dedicated to supporting LBS designers in exploring, filtering, and selecting specific location traces for playback and application testing.

A SCENARIO OF USING TRACEVIZ

Here, we describe a scenario in which a LBS designer would like to use TraceViz to find particular location traces.

David is a LBS designer who is developing an application that recommends promotions to a user based on their current location and recent trajectory. Knowing that such an application would need to intensively respond to location updates at various places in the downtown area, David's team recruits prospective users to collect a large collection of location traces for testing. As the application prototype evolves, David wants to test the prototype with a number of test cases that involve a traveler passing by specific promotions. To find traces that travel specific routes will require David to review all of the traces his team has collected, so he turns to TraceViz.

David launches TraceViz and uses the search box to center the map on a certain restaurant. He enables the brush mode and brushes a route along a street on which a promotion is located. This results in only five traces that pass through the area he brushed on the map. David selects one location trace from this set, and uses it for testing the application.

THE TRACEVIZ INTERFACE

TraceViz consists of three major components, as shown in Figure 1. The *TraceViewer* (Figure 1b) visualizes location traces and allows users to brush to filter traces. Traces are color-coded based on whether they are highlighted, selected, or brushed. Users can hover over a trace to view detailed information in the *TraceInfo Panel* (Figure 1c), and they can click on the trace to select it. The *Control Panel* (Figure 1a) allows filtering based on trace duration and distance, and additionally provides controls for users to select brush modes, adjust brush thickness, and set a brush tolerance threshold. Users can utilize the time and physical length filters location traces by time duration and physical length of the traces in addition to brushing. The brush thickness slider allows users to adjust the coverage of a brush stroke. The tolerance threshold slider controls

whether a brushed trace should be displayed based upon its similarity to the brush stroke. The brush mode buttons allow users to switch among the Reselect, Intersect, and Union modes (described below). Once one has selected a set of location traces, the selected traces can be downloaded in a preferred file format (e.g., GPX) or loaded directly into RePlay [6], a dedicated capture and playback tool.

TraceViz was built using the Flex 4.1 SDK and uses the Google Map API for Flash. It can run on any browser with Flash Player 10 installed. It connects to a MySQL database that contains the traces, and has the ability to import traces in standard file formats such as GPX.

BRUSHING TO EXPLORE AND FILTER TRACES

Brushing is a direct manipulation technique that TraceViz employs to allow users to specify trajectories on a map to filter traces that are "similar" to the brushed stroke. When a brush stroke is drawn on the TraceViewer, we determine which traces are "similar" to the stroke as follows:

1. We identify a set of *candidate traces* by including all traces that have at least one point within the stroke's *candidate area*, which we define as a rectangular area around the stroke that is *2 * brush thickness* pixels larger than the stroke bounds in all directions. The goal of this step is to reduce the number of traces that are subsequently considered while retaining enough information about each trace to be able to determine the degree to which it is aligned with the brush stroke.
2. For each *candidate trace,* we compute the *brush-to-trace similarity* by computing the proportion of *brush points* that are *near* (i.e., within *brush thickness* of) at least one *trace point*. This gives a higher score to traces that are well aligned with the brush stroke throughout the stroke's entire length. It also penalizes traces that have few points that fall within the brush stroke, whether due to misalignment or due to sparse data.
3. We also compute the *trace-to-brush similarity* for each trace by computing the proportion of *trace points* that are *near* at least one *brush point.* This gives a higher score to traces whose nearby points lie mostly within the brush area and penalizes traces whose trajectories diverge from that of the stroke. While in most cases the *brush-to-trace* and *trace-to-brush* scores are redundant, both are needed to deal with cases where the trace and brush point densities differ.
4. Finally, we compute the *overall similarity* by averaging the *trace-to-brush* and *brush-to-trace* similarity scores. Note that all traces that lie outside the *candidate area* are assigned an *overall similarity* of zero.

Each trace is then rendered on the map according to its similarity score: first, all traces with similarity scores below the user-defined *tolerance threshold* are given alpha values of zero, and all other traces are given alpha values proportional to their similarity score.

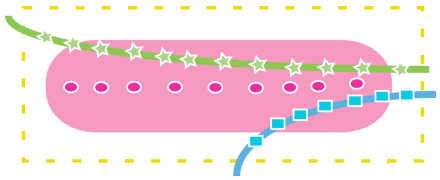

Figure 2. When a brush stroke is drawn, TraceViz computes the "similarity" of all nearby traces to the stroke. Here, two *candidate traces* intersect the *candidate area* (dashed yellow line). The top trace (green stars) is found to be more similar than the bottom trace (blue squares) because more of its points lie within the brush stroke region (pink oval).

As an example, consider the situation depicted in Figure 2. Trace 1 (green stars) is assigned a *brush-to-trace* similarity of 1 since all of the *brush points* are near to at least one *trace point*. It receives a *trace-to-brush* similarity of 0.83 since 10 of 12 *candidate points* are near at least one *brush point.* The *overall similarity* is therefore 0.92. Trace 2 (blue squares), on the other hand, receives a *brush-to-trace* similarity of 0.44 since 4 of 9 *brush points* are *near* at least one *trace point,* and a *trace-to-brush* similarity of 0.71 since 5 of 7 *trace points* are near a *brush point.* Trace 2's *overall similarity* is thus 0.58.

In order to give LBS designers additional control, TraceViz provides three brush modes—*Reselect*, *Intersect*, and *Union*. In Reselect mode, every stroke generates a new result. In Intersect mode (shown in Figure 3), each stroke after the first refines the filter to show only traces that pass through all strokes, whereas in Union mode each stroke widens the filter to include traces that pass through *any* stroke. As an example, we return to our earlier scenario to illustrate the use of Intersect mode:

David wants to find a trace passing by both a coffee house and a bookstore that are on two different streets. He uses the Intersect mode to brush two strokes near the coffee house and the bookstore, respectively. However, he finds that the filtered traces are still many and overlap one another. As a result, he brushes the third stroke on another street to refine the filtered results. Now David can easily distinguish the traces and select them for testing.

USER STUDY

To observe whether and to what extent TraceViz can help LBS designers efficiently select location traces, we conducted a preliminary usability evaluation. We recruited eight people with experience in designing or developing mobile applications to participate and asked them to use TraceViz to perform four tasks in which they selected traces to load into RePlay [6]. Their goal was to test aspects of a sample LBS called Here & Now (H&N), which is a location-based advertising application we developed for the purpose of testing TraceViz and related tools. H&N allows merchants to create promotions that are delivered to mobile customers and allows customers to see the promotions nearby their current location. Customers can adjust the "notification range" within which they receive information

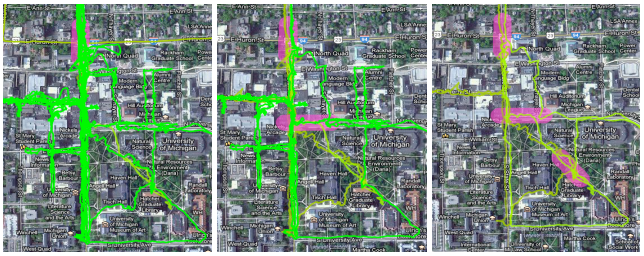

Figure 3. The Intersect brush mode allows an LBS developer to refine a filter by adding additional brush strokes, as shown here from left to right.

about promotions. Our tasks were all based on finding traces that would be suitable for testing this feature.

In the first task, participants were asked to practice selecting a trace by brushing. In the second task, they needed to find a trace that allowed testing different display ranges (0.5 km, 1km, and 2km) and show that H&N respected users' preferences in all cases. In the third task, participants had to find traces that passed by at least two active promotions. Finally, in task 4, participants needed to find two traces that approached a promotion from different directions in order to show H&N working with multiple simultaneous users. We provided each participant with 200 GPS traces collected in Ann Arbor, Michigan, USA (the city where our study took place). All participants received a demonstration of TraceViz, RePlay, and H&N at the beginning of the session. Upon completing the tasks they were asked about their reflections on using TraceViz. Participants were encouraged to solve the tasks in any way they wished, and were not directed to use particular features of TraceViz. Video and audio for all sessions was captured, along with detailed session notes, and these data were reviewed to assess and interpret task success, critical incidents, and participant satisfaction.

Seven of the eight participants were able to finish all four tasks with little or no assistance. Most participants were able to find suitable traces within one or two attempts for each task. This suggests that TraceViz as a whole is able to support LBS designers' in efficiently finding and selecting traces for testing a LBS. Moreover, participants developed several different strategies of using brushing for finding suitable traces. For example, while some participants used the Intersect mode for finding traces passing by two specific areas, other participants used it simply for reducing the number of traces on the map.

However, we also uncovered several shortcomings to be addressed in future versions of TraceViz. Some shortcomings were basic usability problems, such as confusion about the brush mode names and difficulties switching between brushing, selecting traces, and panning the map. Additionally, most participants struggled when choosing a trace with unexpected characteristics, including changes in direction and speed or poor signal quality. For instance, P2 selected a trace with sparse location records (probably due to poor GPS signal), and it took her a long time to accomplish one of the tasks as a result. Improving TraceViz to provide more detailed information about traces, such as speed, signal quality, and direction of travel would likely help with this problem. We also observed that participants had trouble understanding a trace's overall trajectory once they had zoomed the map in too far. Better overviews in the TraceInfo Panel could help here. Finally, sometimes participants still encountered difficulties selecting overlapping traces even after brushing and using other filter controls. This suggests that more precise selection capabilities may be needed for especially dense data sets. We plan to address these issues in future work.

CONCLUSION

In this paper we have presented and evaluated TraceViz, a novel tool that employs "brushing" to support LBS designers in exploring, filtering and selecting GPS location traces from large data sets. TraceViz supports Reselect, Intersect, and Union modes to enable LBS designers to more flexibly explore and filter location traces based on trajectories in order to meet various testing requirements. We believe that with the support of TraceViz, LBS designers will have more incentive to test their LBS with a larger variety of location traces, which will in turn help the designers produce higher quality designs.

ACKNOWLEDGMENTS

We thank Eytan Adar, Mark Ackerman, and members of the Interaction Ecologies group for helpful comments on this work. We also thank our study participants.

REFERENCES

1. Andrienko, G., et al. Interactive visual clustering of large collections of trajectories. In Proc. *VAST 2009*, 3-10.
2. de Silva, G.C., Yamasaki, T., and Aizawa, K. Sketch-based spatial queries for retrieving human locomotion patterns from continuously archived GPS data. *IEEE Transactions on Multimedia 11*, 7 (2009), 1240 -1253.
3. Girardin, F., et al. Digital Footprinting: Uncovering Tourists with User-Generated Content. *IEEE Pervasive Computing 7*, 4 (2008), 36-43.
4. Hurter, C., Tissoires, B., and Conversy, S. FromDaDy: Spreading aircraft trajectories across views to support iterative queries. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1017–1024.
5. Monmonier, M. Geographic brushing: Enhancing exploratory analysis of the scatterplot matrix. *Geographical Analysis 21*, 1 (1989), 81–84.
6. Newman, M.W., et al. Bringing the field into the lab: supporting capture and replay of contextual data for the design of context-aware applications. In Proc. *UIST 2010*, 105-108.
7. Raper, J., et al. Applications of location–based services: a selected review. *Journal of Location Based Services 1*, 2 (2007), 89–111.