A Framework for Personalizing Action History Viewer

Masaki Ito
niya@ht.sfc.keio.ac.jp

Jin Nakazawa
jin@ht.sfc.keio.ac.jp

Hideyuki Tokuda
hxt@ht.sfc.keio.ac.jp

Graduate School of Media and Governance
Keio University
5322, Endo, Fujisawa, Kanagawa, Japan

ABSTRACT

This paper presents a programmable analysis and visualization framework for action histories, called mPATH framework. In ubiquitous computing environment, it is possible to infer human activities through various sensors and accumulate them. Visualization of such human activities is one of the key issues in terms of memory and sharing our experiences, since it acts as a memory assist when we recall, talk about, and report what we did in the past. However, current approaches for analysis and visualization are designed for a specific use, and therefore can not be applied to diverse use. Our approach provides users with programmability by a visual language environment for analyzing and visualizing the action histories. The framework includes icons representing data sources of action histories, analysis filters, and viewers. By composing them, users can create their own action history viewers. We also demonstrated several applications on the framework. The applications show the flexibility of creating action history viewers on the mPATH framework.

Keywords

Action history, visualization, visual language

INTRODUCTION

In the ubiquitous computing environment where computers and sensors are embedded in our surroundings, it will be possible to recognize our action and record it. From the accumulated action history, we will be able to get highly abstracted context information of the human activity. These information are used to develop context-aware applications, and also used to provide us with useful information. Well presented action histories help our life such as retrieving memory and sharing our experiences.

Several representation applications of the action history have also proposed[3][6][8][9][10][11]. These systems represent user’s activity to provide functionalities of navigation and indexed action histories. However these representations are designed for a specific use, and hence users can not customize them to acquire personalized view of their action histories. For example, though PEPYS[8] can organize human action histories based on the location- and time-axis, it can not handle additional information, such as images, related to an action history item. Activity Compass[9] provides a navigation functionality based on a location track analysis, which also lacks diversity of location history analysis. The action history viewers, therefore, should provide users with personalized views for enabling them to analyze, recall, talk about, and report their action histories.

In this paper, we propose a new framework for creating the personalized action history viewer. The framework provides users with programmability by a visual language environment for analyzing and visualizing the action histories. The framework includes icons representing data sources of action histories, analysis filters, and viewers. By composing them, users can create their own action history viewers.

This paper is organized as follows. In the next section, we introduce scenarios where various visualization technique for analyzing action history. The third section shows current techniques for analysis and visualization and clarify the requirements. In the 4th section, we introduce mPATH framework as a framework of personalizing action history viewer. Next four sections introduce the usage of the mPATH framework and show several applications. We evaluated the system in the following session and introduce related works. In the final session, we conclude this paper and suggest future work.

SCENARIOS

We introduce two scenarios where visualization method are shared and easily developed by users. These scenarios show usage of action history viewers with a personalizing function. The feature helps our communication and the deep understanding of past activity in the scenarios.

Navigation of a Travel Memory

Last week, Alice has traveled Kyoto, Japan. When she arrived at Kyoto station, she borrowed a PDA with GPS as a guide for sightseeing. While she was in Kyoto, the PDA assisted her in planning her travel and gave her a guidance at a tourist attraction. When she left Kyoto, she returned the PDA and received a small memory
card in which her location track data, names of tourist attractions which she visited, and photos she took were recorded.

Today she is talking about her travel with her boyfriend, Bob. She inserted the memory card to her PC and showed a map of Kyoto which was overlaid with lines of location track data. She started talking on her experience from the beginning of her travel, but immediately found the map were not designed to represent temporal aspects of her travel. She searched for visualization methods of travel experience and downloaded them.

One method analyzed her travel log and calculated weights of each tourist attraction she visited by her walking speed and number of pictures. The method visualized the map of Kyoto with distortion which stands for the weights, she could intuitively know her travel.

She thought shopping is also important to calculate the weights, she changed the parameters of analysis and generated a map which highly reflect her impression. The map realized smooth understanding of her travel for him.

Development of Analysis and Visualization Method Now, Bob wants to go to Kyoto. He asked her to go with him, but she refused because she has just been there. He then decided to show the attraction of Kyoto she did not know.

Since he is an amateur programmer, he decided to develop a visualization method in which attractive places where she did not visit were emphasized. He at first searches for a web guide of Kyoto in which tourist attractions are ranked and contains many pictures. Then he developed an algorithm to find attractions she did not visit by comparing the web guide with her tour data. He designed to visualize a map of Kyoto with many photos of the attractions.

He uploaded his algorism and asked her to download and apply it to her memory. She noticed unknown attractions in Kyoto and decided to visit again.

VISUALIZATION OF ACTION HISTORY

In this section, we define action history, and mention current techniques of visualization and analysis of action history. Then we clarify requirements of the visualization system.

In this paper, an action history is an aggregated form of information which contains location, date and description of action about a certain person. Location track data obtained by GPS is one example of action history. Digital photo data is also an action history if it contains a time stamp and location information as its meta information.

Current Visualization

There are several visualization methods for representing action history. We introduce some of them and argue their features.

Text-based Visualization Text-based visualization is a simple technique to represent daily and special experiences. Without machinery, some people keep diaries not to forget daily events, and in some situations, to share a secret with a intimate friend by exchanging a diary book. Text-based style has no restriction of a format and contents, therefore we can easily represent various experiences in the style. However it is difficult to find certain information with a specific point of view from a diary.

List List is a structured format of text which shows specific aspect of experiences in order. A chronology is one example of list which shows a temporal aspect of history. This style helps intuitive understanding of a specific feature of action history such as time, event, and name of place.

PEPYS[8] represents user’s activity as a list in temporal order. We can know temporal context of each action. However, it is difficult to know spatial aspect of the action since rooms which he or she was in are represented only as names.

Map-based Visualization A map, which is widely used to represent geographic information, can be also used to represent a spatial aspect of action history. Overlaying a ready-made map with points and lines which suggest certain action history is a popular method for representing action history. We can understand an action history in a geographic context easily by using such a map. Map-based visualization is utilized in several researches[9][11]. However, a ready-made map contains only common objects like restaurants, gas stations and hotels, and not enough to show personal experiences.

3D Map Visualization Especially for hikers and climbers who are logging their tracks by handy GPS devices, several applications in which mountains and valleys are shown as 3D graphics, and they are overlaid with lines of the tracks. KASHMIR 3D[10] is widely used in Japan, and Wissenbach Map3D[3] is also developed for the same purpose. These applications utilize digital elevation model of a certain area for creating terrain model.

Photo-based Visualization Photographs taken by a user represent his or her interest during his or her activity like travel. Simply placing many thumbnails of photos on a screen is widely utilized technique. However, spatial and temporal aspects of photos disappear in this visualization.

STAMP[6] represents spacial relationship of each photo by linking the same object in two photos. We can brows photographs by following the links by a mouse. This method simultaneously visualize user’s points of view and spatial structure.

Analysis of Action History

As most of the visualization technique represent only a few aspects of action history, analysis method of action history to extract certain aspects is also important in a visualiza-
tion process. Currently, analysis and visualization are tightly combining. We argue them separately in this paper.

Time and space are basic aspects of action history. We often utilize them to order action history and as clues of retrieving certain history. Most of the visualization technique are designed to represent both or either of them.

By analyzing action history, we can acquire highly abstracted information like a frequency of visiting a certain place, a daily pattern of a movement, and user’s interests and preferences. There are several researches to analyze location track data captured by GPS and extract such information [1][9][12]. Some of these researches utilize additional geographic information to detect an activity in a certain place.

To utilize action history as an assistant of our memory and communication, we should understand various aspects of action history as the scenarios show. However, current visualization systems are designed for specific use of action history. To represent action history in various aspects, flexible analysis and visualization features are required.

Requirements
For flexible programming of visualization mentioned in the scenario, following features are required to the system.

Flexibility of Data Input The system must treat various types of data available in the ubiquitous computing environment simultaneously. Action histories are characterized by their description of location, contents of what we did there, and the way of acquisition of the data.

The system also need to treat geographic information for analysis of action history. A digital map and the Yellow Pages with address are examples of geographic information.

Providing Programmability of Analysis and Visualization For all users of the system, it must be easy to create their original visualization method. For skillful users, the system must provide a flexible programmability. Even for unskilled users, the system must provide possibility of changing a visualization method.

Sharing existing methods, which are programmed by third parties, also increases programmability of the system. Using existing method reduces cost of creating new analysis and visualization method.

mPATH FRAMEWORK
We developed a programmable analyzing and visualizing framework for action history named mPATH framework. In this section we introduce the features and implementation of the mPATH framework.

Approach
The followings are the approaches taken by the mPATH framework to accomplish aforementioned flexibility.

Component Based Architecture For reducing the cost of developing a new analysis and visualization method, we divided the method into several components. We defined three types of component, data source, viewer and filter. Users can create visualization method by combining existing components.

Standardization of Internal Data We defined a unified type of data in the system. The description of location, date and other information are unified. When we input action history or geographic information, they must be converted into single type of data. The type of data is used to exchange action history between components. Since data type is unified, we can easily design a component for several action history. The feature also realize flexible combination of components.

Using Data Flow Style Visual Language To control combination of components, we provide a visual programming system of data flow style. Visual programming reduces difficulty of creation of original visualization method by combining components.

Implementation
We implemented a prototype of mPATH framework with Java. Our system is implemented as a GUI application and consists of 14,000 lines of Java language. Figure 1 shows a screen shot of the system.

![Figure 1: Screen shot of the system](image.png)

In the current implementation, we mainly focus on realizing visual programming for data analysis and visualization. This version works as a platform for creating analysis method by data flow style programming. Exporting and importing of program function is still under construction. We are planning to use XML to exchange them.

Experiment
Since June 2003, Ito, the first author of this paper, has been carrying a “Garmin eTrex Legend”[5], a handy GPS receiver.
Table 1 shows the amount of captured data. While the experiment, he took pictures as action histories using a digital camera. The track data of the handy GPS and taken pictures are used as a data of development and evaluation of this system.

### Table 1: Captured Data of GPS and Digital Camera

<table>
<thead>
<tr>
<th>Date</th>
<th>size(byte)</th>
<th>track</th>
<th>point</th>
<th>picture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jun. 2003</td>
<td>321,310</td>
<td>286</td>
<td>5,893</td>
<td>515</td>
</tr>
<tr>
<td>Jul. 2003</td>
<td>294,518</td>
<td>297</td>
<td>5,336</td>
<td>131</td>
</tr>
<tr>
<td>Aug. 2003</td>
<td>473,739</td>
<td>412</td>
<td>8,676</td>
<td>218</td>
</tr>
<tr>
<td>Sep. 2003</td>
<td>365,855</td>
<td>298</td>
<td>6,730</td>
<td>37</td>
</tr>
<tr>
<td>Nov. 2003</td>
<td>193,772</td>
<td>153</td>
<td>3,573</td>
<td>54</td>
</tr>
<tr>
<td>Dec. 2003</td>
<td>205,621</td>
<td>208</td>
<td>3,727</td>
<td>7</td>
</tr>
<tr>
<td>Jan. 2004</td>
<td>278,071</td>
<td>215</td>
<td>5,135</td>
<td>108</td>
</tr>
<tr>
<td>Feb. 2004</td>
<td>292,319</td>
<td>231</td>
<td>5,392</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td>2,732,392</td>
<td>2,387</td>
<td>50,070</td>
<td>1,417</td>
</tr>
<tr>
<td>Average</td>
<td>303,599</td>
<td>265</td>
<td>5,563</td>
<td>157</td>
</tr>
</tbody>
</table>

**COMPONENTS FOR mPATH FRAMEWORK**

We developed several components for mPATH framework. In this section, we classify them into data source, filter and viewer, and introduce components we developed.

**Data Source**

To input various action history, we developed several components as data sources. These components access action history and transform the data into standard data format.

**GPS Location Track Data Source**

GPS Location Track Data Source access files of location track information or a GPS device through an RS-232C interface. Acquired data are transformed into a group of points where the actor passed by.

**Photo Data Source**

Photo data source deals with image files taken by a digital camera. By reading time stamp and location in EXIF[7] information of the files, the components generate an action history of “taking pictures” in the standard data format.

The component can accept location track data. By matching timestamp of photos with location track data, it estimates location of pictures.

**Map Data Source**

Map data source inputs map data in a vector format such as points of station, lines of road and polygons of buildings. Users can detect detail of action history by comparing it with the map data. Since the data was designed for GIS, we can get generic map by rendering them and use the map as a background of visualization.

**Filters**

Components of filter have one or more inputs and outputs. They input data of standard format and output the result of processing in the same format.

**Time Filter**

Time filter extracts action histories during a specified term. Through the GUI of the filter, we can change the term. The operation immediately affects the output on the viewer.

**Speed Filter**

Speed filter filters data by its speed. It is useful especially for infer transportation from location track data.

**Formalize Filter**

Formalize filter classifies a group of point data into movements and stops, and outputs as track and point data by two separated sockets. Users can change threshold of time to detect actor’s stop.

**Matching Filter**

Matching filter has two input sockets: one for the map data and the other for a geographic coordinate. The matching filter calculates the name of the specified coordinate from the specified map.

**Count Filter**

Inside count filter, geographical region are divided into a grid. The filter counts input data in every grid. Users can know how many times he or she visited a certain point, i.e. a weight of the point, in an action history.

**Viewer**

We developed three viewers to visualize action history.

**Normal Map Viewer**

To visualize spatial aspect of action history, we developed normal map viewer. In this viewer, every input are ordered by the geographic coordinates, so that generic map-like visualization is realized. This viewer accepts multiple data and overlays them. Figure 2 shows an image of normal map viewer.

**Table Viewer**

We used table type viewer shown in Figure 3 to visualize data output by the matching filter. In the left column, the name of the place he visited are listed, in center,
time of stay are shown. In the right column, the date the user begin to stay are shown.

Figure 3: Table Viewer

Weight Map Viewer To represent highly abstracted context of action, we developed weight map viewer. The viewer accepts weight of regions in addition to normal action history or geographic information. This viewer visualize the weight as a color or scale of each region as shown in Figure 4.

Figure 4: Weight Map

VISUAL PROGRAMMING

In this section we introduce a visual programming manner of mPATH system by creating map like viewer of action history. The main window of the mPATH framework consists of mainly two windows, one is a pallette and the other is a canvas. Figure 5 shows the detail.

In the pallette, components of data sources, viewers and filters are registered as icons. By dragging and dropping of the icon, a component is copied and registered to the canvas.

In the canvas, every data flow is shown as lines from left to right. Each icon of component has sockets of data, Right socket means an output of the data, and left socket means an input. Icons only with right socket are data sources, and icons without right socket are viewers. Icons with both reft and right socket are components of analysis method and work as filter of data. By clicking two icons on canvas, two components can be connected and disconnected.

Figure 5: Visual Programming Window

Figure 6: Example of Visual Programming

Figure 6-(1) is a simple visualization of location track data of GPS. In this program, GPS data are input to normalize filter. Only track data are input to a normal map viewer and shown in the geographic coordinates. The result of visualization is Figure 2-(1). When we connect both track and point socket to the viewer, we can see stop point data with track data as shown in Figure 2-(2). The operation on the canvas is immediately reflected on the viewer, therefore an interactive operation is accomplished.

By insertion of time filter between GPS data source and formalize filter as shown in Figure 6-(2), we can control the term of location track data.

To visualize the detail of location track data, we use a map as a background. Figure 2-(3) is a result of connection between map data source and normal map viewer directory. By inputting the output of map data source to the viewer of track data as Figure 6-(3), we can see track data with normal data as Figure 2-(4).

We can join data of two inputs by comparing geographic coordinates of each input by a matching filter. By utilizing this
filter, we can acquire a name of a stop point from a map. Figure 6-(4) is a program which utilize matching filter to acquire names of stop points by inputting a map data and a formalized filter. In this program, we visualize the output as a list by using table viewer shown in Figure 3.

**DEVELOPMENT OF A DATA SOURCE, A FILTER AND A VIEWER**

If ready-made data sources, filters or viewers do not satisfy users, they can also develop original one with Java code. We provide a skeleton of them, users can create a new data source, filter and viewer by extending four functions in the skeleton.

Inside the mPATH framework, data sources, filters and viewers are designed as the same object named ActionFilter. An ActionFilter can have any number of input and output connector. If an ActionFilter is equipped with no input connector, it works as a data source. If an ActionFilter is equipped with no output connector, it works as a viewer. An ActionFilter with both input and output connector works as a filter.

Since a data transfer mechanism in the mPATH framework is designed as a demand-driven style, filtering logic is implemented in a function which returns result of the filter. A messaging mechanism to notify change of upper ActionFilter is equipped to realize data-driven analysis. When the lowest ActionFilter receives a change event, it demands newer result of upper ActionFilters and accomplishes data-driven analysis.

Following functions are prepared for developers.

1. **getActionElement(GeoShape area):**
   - This function was called when analysis result is required by lower ActionFilters. An area information is given as an argument, the function must return all analyzed result in a format of ActionElement.
   - When developing a filter, a developer implements logic of the filter in the function. When developing a data source, this function is utilized to acquire action history and form them into the unified type of data.

2. **afterConnectFrom(ActionFilter fromFilter):**
   - This function is called when other ActionFilter is connected to upper Socket. A connected ActionFilter is noticed as an argument.
   - When developing a viewer, it is needed to change the output of the viewer when this function is called.

3. **afterDisconnectFrom(ActionFilter fromFilter):**
   - This function is called when connected upper ActionFilter is disconnected. A disconnected ActionFilter is noticed as an argument.
   - When developing a viewer, it is needed to erase the output of the viewer when this function is called.

4. **preNotification(ActionFilter fromFilter):**
   - This function is called when the state of upper ActionFilter given as an argument is changed. After processing this function, the change event is transmitted to lower ActionFilters.
   - When developing a viewer, it is needed to refresh the output of the viewer in this function.

**APPLICATIONS**

In addition to normal map viewer application mentioned as an example, we developed two applications on the mPATH framework. First one is listing points of interest system by analyzing stay stop in a certain place. Second one is a visualization system of travel activity focusing on traveler’s interests on the places.

**Listing Points of Interest System**

We developed extraction system of user’s interest point on the mPATH framework. An interest point is a place where user did something or stayed for a long time. We also detect name of the point using matching filter. In this system, we visualized interest points in a table. Figure 3 shows the result of the visualization.

**Visualization with Weight**

We developed an weigh method to classify regions by user’s activity. In our algorithm, we at first divide the region into small grids. In each grid, we count the number of times we visited, the number of times we took pictures, and the number of times we shopped. Then we add each value and obtain the weight of each cell as a number.

We implemented this algorithm on the mPATH framework, and developed a visualization system reflecting the weight of the region. Figure 4 shows an example. In the example, we weighed each cell mainly by the number of pictures actor’s took, and the weight of each cell is visualized as a scale. We realized showing photo by clicking on the map, this system can be used as a photo viewer.

**EVALUATION**

In this section we evaluate the performance of the mPATH framework. It shows if the response of the system is enough to develop interactive development of visualization methods. It can also be used to find functions which should be improved.

For measuring the performance, we used the environment shown in table 2 and used data shown in table 3.

<table>
<thead>
<tr>
<th>Table 2: Evaluation Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
</tr>
<tr>
<td><strong>memory</strong></td>
</tr>
<tr>
<td><strong>OS</strong></td>
</tr>
<tr>
<td><strong>JDK</strong></td>
</tr>
</tbody>
</table>
Table 3: Data

<table>
<thead>
<tr>
<th>Type</th>
<th>GPS location tracking</th>
<th>Map Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>Jun. 2003 – Nov. 2003</td>
<td>Fujisawa city</td>
</tr>
<tr>
<td>size</td>
<td>1,956,381 byte</td>
<td>9,261,000 byte</td>
</tr>
</tbody>
</table>

**Overhead of component architecture**

We measured the performance of the component architecture, therefore the architecture seemed to be an overhead compared to a hard-coding implementation. We measured simple visualization method with several time filters, in which we changed the number of the time filter.

![Figure 7: Measurement of the Overhead](image)

**Figure 7: Measurement of the Overhead**

Figure 7 shows the result of the measurement. While we increased time filter from zero to 15, the growth of the time is small. This result shows that the overhead of connecting analysis components is small.

**Evaluation of the Performance of the Visualization on the System**

We measured the performance of a visualization system constructed on the mPATH framework. We used simple visualization application in which location track data and map data are rendered on a same window, and measured the time required for rendering. Figure 8 shows the result. It also shows the result of rendering single location or map.

Lowest case of rendering time is about 800ms, and in the case of small scale maps, it takes more than 1500ms. In the overlaid case, the rendering time of the map are two times the single rendering of the map, and can be reduced.

**RELATED WORKS**

We introduce several researches and products as related work of mPATH framework. These systems do not treat action history, but provide flexibility of data analysis and representation by visual language manner.

**DFQL**

To improve query language of databases, various graphical query languages are proposed instead of text based language like SQL. DFQL[4] is one of the graphical query languages for the use in scientific database. DFQL uses data flow language and enable analysis and visualization in addition to data retrieval which general graphical query languages focus on.

**Max/MSP**

Max/MSP[2] is a visual programming system for midi and audio signal. The system enables interactive processing of music stream with a graphical data flow language and is used to create electronic musical instruments or effectors with original sound algorism, and interactive media systems.

The visual language is used by many creator of electronic media and fruits of the programming are exchanged widely on the Internet. Since modules of the visual language can be developed with C language, many original modules are also distributed.

**CONCLUSION**

In this paper, we presented a programmable analysis and visualization framework for action histories, called mPATH framework. The mPATH framework provides data flow visual language, and enable flexible and interactive analysis by connecting analysis components through mouse operation. By component architecture, the framework enable providing various visualizations which represent various aspects of action history. We implemented the mPATH framework with Java language, and demonstrate constructing various viewer applications. We also evaluated performance of the framework, and proved its interactivity. We are planning to extend mPATH framework especially focusing on following issues.

**Implementation of a Sharing Mechanism**

We are now implementing a mechanism to share a program on the mPATH framework. We designed XML based description as a file format of programs on the mPATH framework. We will implement functions to import and export the XML. We are also planning to provide a server on the Internet to upload and download the XML files.
Enable Visual Programming of Parameters of Modules  

We will extend the visual language and enable visualization of parameters of each modules. We will enable parameters of models to be treated as a modules in the system. This feature will realize coordination of each module and enable more complex analysis and visualization.

REFERENCES


