Journal of Mobile Multimedia, Vol. 1, No. 3 (2005) 255–269

mPATH: A SOFTWARE FRAMEWORK FOR INTERACTIVE VISUALIZATION OF BEHAVIOR HISTORY

MASAKI ITO
Graduate School of Media and Governance, Keio University
5322, Endo, Fujisawa, Kanagawa, 252-8520, Japan
niya@ht.sfc.keio.ac.jp

JIN NAKAZAWA
Graduate School of Media and Governance, Keio University
5322, Endo, Fujisawa, Kanagawa, 252-8520, Japan
jin@ht.sfc.keio.ac.jp

HIDEYUKI TOKUDA
Faculty of Environmental Information, Keio University
Graduate School of Media and Governance, Keio University
5322, Endo, Fujisawa, Kanagawa, 252-8520, Japan
hxt@ht.sfc.keio.ac.jp

Received May 12, 2005
Revised Aug 15, 2005

This paper presents an interactive analysis and visualization framework for behavior histories, called mPATH framework. In ubiquitous computing environment, it is possible to infer human activities through various sensors and accumulation of their data. 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 interface for analyzing and visualizing the behavior histories. The framework includes icons representing data sources of behavior histories, analysis filters, and viewers. By composing them, users can create their own analysis method of behavior histories. We also demonstrate several visualizations on the framework. The visualizations show the flexibility of creating behavior history viewers on the mPATH framework.

Keywords: Behavior History, Visual Programming, Information Visualization, GIS, Interactive System

1 Introduction

In ubiquitous computing environment where computers and sensors are embedded in our surroundings and assist our life, cognition of human behavior will be possible. Today, sensing technology is one of the hottest topics for ubiquitous computing researchers. In addition to the advancement of sensing technologies, various technologies for organizing a network by sensors have developed[1][2]. Sensed data are exchanged and processed within the network, then they work greater than the sum of each sensor. Using sensor networks, we can acquire

not only data of a certain place such as temperature, humidity, noise level or light level, but also data of moving objects, such as their track, change of status while moving and even human activities.

Technologies for mobile computing[3][4][5] are also useful to sense human activities and environmental information. Sensors equipped with a mobile device acquire various information of the user such as a location, movement orientation and a physiological status by alone or coordination with sensors in surroundings[6][7].

Inferring highly abstracted behavior of a user and creating predictive model of user's behavior from the sensed data are also hot challenges in ubiquitous computing research. There are many researches which aim to guess user's context and provide adaptive services. Several context processing technologies are proposed for the purpose[8][9].

In addition to processing sensor data in a certain time, storing and utilizing an accumulation of behavior data brings us a possibility to develop new applications[10]. A well presented behavior history helps human activities 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[11][12]. For example, displaying user's shopping log reminds him or her of goods which he or she intended to buy, and helps explanation of his or her intention of shopping to his or her company. Analysis techniques of behavior history are important to find useful information from it.

There are already various proposed methods for analyzing accumulated behavior history. However these methods are designed for a specific use, and hence users may not acquire applicable information. The analysis method of behavior history, therefore, should be flexible enough to find various information from behavior history.

In this paper, we presents a programmable analysis and visualization framework for behavior histories, called mPATH framework. In the mPATH framework, analysis methods are modularized and we can create various analysis methods using visual programming language. The framework allow every developers and users to mine behavior history intuitively and flexibly.

This paper is organized as follows. In the next section, we argue problems of visualization of behavior history. In section 3, we introduce mPATH framework which realizes flexible analysis of behavior history using visual language. We also demonstrate several filters on the mPATH framework in section 4. In section 5, we introduce applications of the mPATH framework. We evaluate the framework in section 6 and introduce related works in section 7. In the final session, we conclude this paper.

2 Visualization of Behavior History

In this paper, behavior history is an aggregated form of information which contains location, date and description of behavior about a certain person. Movement data obtained by a GPS receiver is an example of the behavior history. A digital photo image is also behavior history if it contains a time stamp and a location information as its meta-data.

Well presented behavior histories help our memory and communication. We often remember our past behavior in daily life. For example, we often plan what we will do for the weekend remembering stores and places which attracted us in the past. Also, we often talk with people remembering our past experiences. There are several researches to assist memory

and communication by visualizing past behaviors[13][11][12].

When we remember our past behaviors, we do not merely remember them chronologically, but remember them selecting and organizing the informations that we need then. For instance, when you tell about the travel that you had, you will tell the places and foods that you were interested in. Also, you might focus on the places where you visited with the listener. Thus, which information you need in your past behavior varies depending on the situation.

Behavior history also useful in modeling user's activity and predicting next action. In many context-aware application, researchers uses behavior history for adaptive behavior of the systems. Such applications infer user's interest and purpose by analyzing behavior histories. While developing a context-aware application, developer need to select and organize for a certain application.

2.1 Behavior History Analysis for Visualization

We focus on the analysis process of behavior histories to find interesting data and organize them when we develop memory assist applications and context-aware applications. Following functions are required for analysis system of behavior history.

Support of Temporal and Spatial Analysis

Since behavior histories are characterized as their location and time parameter, supporting temporal and spatial analysis is required in addition to generic numeric analysis. Selection of data within a certain location or period is one example. Joining multiple data using a certain time or location as a key is another example. These analysis assist in finding characteristics in behavior histories.

Support of Trial and Error Analysis

The diversity and irregularity of captured data from sensors require heuristic development of analysis method. Users who analyze behavior history need to adjust algorism and parameters for a certain history data. Thus, the system should be programmable to support such trial and error development of analysis method.

Reusability of Analysis Method

Since there are various behavior histories, developed analysis method specialized to a certain behavior history should be reused to other histories. For example, analysis method for finding interesting points from a shopping history should be applied to photography history. Analysis methods, therefore, are independent of a certain history format.

3 An Interactive Behavior History Mining Framework

In this paper, we propose an interactive behavior history analysis framework called mPATH framework. We can input various behavior histories to the framework and analyze in various ways on it. The mPATH framework provides a visual programming interface, so that users can develop various analysis method intuitively through GUI. Figure 1 shows the screen shot of the system.

3.1 Features of the mPATH Framework

The following are features of the mPATH framework.

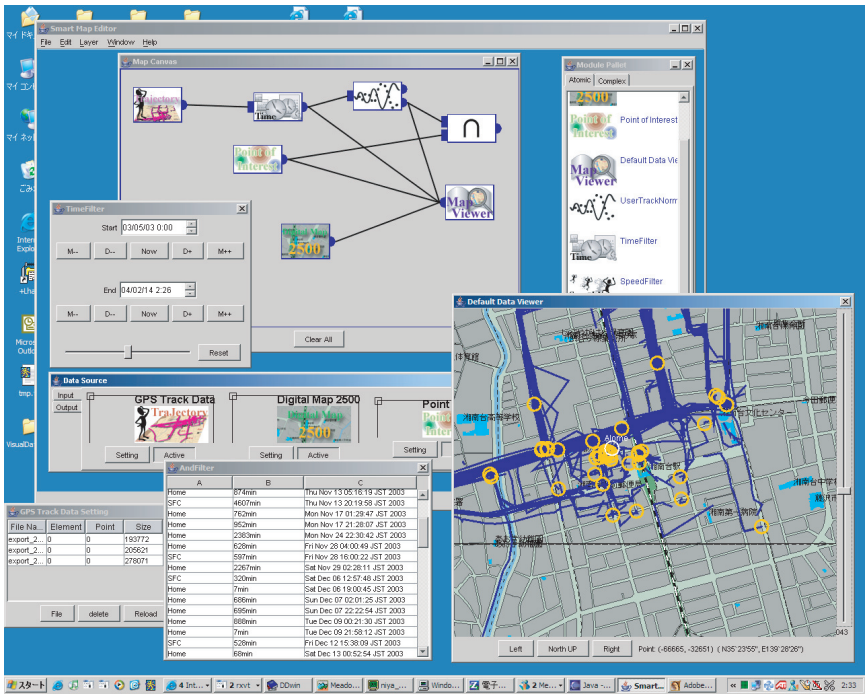

Fig. 1. mPATH Framework

Component-based Architecture

For supporting programmability and reusability, we divided the method into several components and enabled development of an analysis method by combining several components like filtering data, counting data and classifying data.

Interactive Visual Programming

We provide data-flow visual programming interface to combine multiple analysis components. Structure of analysis method is shown visually and users can modify it by simple mouse operation. User's operation is immediately reflected to the system. Therefore users can easily develop original analysis method for a certain visualization.

Unified Behavior History Format

We defined unified internal data format of behavior history and geographic information. Since all internal data can be accessed by the unified way, we can easily reuse analysis method to various behavior history.

3.2 System Architecture

To prototype the system rapidly, we designed mPATH framework as a general GUI application, and developed the system using Java. The system, therefore, is designed as a collection of functions which are called by a GUI library when a mouse or a keyboard is operated.

We adopted data-flow visual programming style [14] to represent internal status of the system. We can construct data-flow programming system as a data-driven system or a demand-

driven system. In the former style, source modules push data to the destination. In this style, unnecessary data are also processed, thus we chose the latter style, on which a destination module requests necessary data to the upper modules. The requested module also request to the upper modules if necessary, and return data with processing at the module. Finally source data are passed to the destination with appropriate processing.

Demand-driven system, however, does not immediately reflect change of data, data-flow, and parameter in filters. Therefore, we also implemented mechanism to notify change in an upper module to the lower ones. When the destination module receives the message, it requests new results to upper modules. Then the system works interactively with user's operation.

When requesting data to an upper module, a lower module gives a rectangle that represents a needed area. An upper module does not need to return data out of the area. It reduces data to be processed inside the system. If the data in an upper module are generated logically, the area becomes more important. It cannot stop returning data without the area, since the data are not aggregation of finite data, but results of certain calculation.

Figure 2 shows a whole architecture of the system. As the figure 2 shows, this system consists of following four parts: Visual Programming Interface, mPATH Core, Module Manager and mPATH Modules. Following are the details of each part.

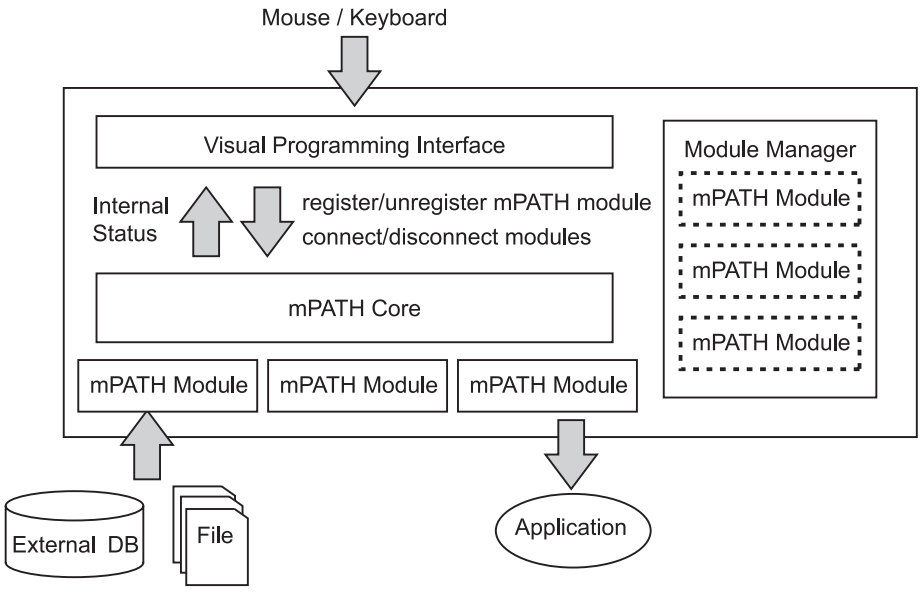

Fig. 2. System Architecture

Visual Programming Interface

Visual programming interface part shows current internal status visually, and accept user's commands to change the status. On the interface, procedure of visualizing data is shown as a collection of icons connected by lines on which data are flowing from the left to the right. An icon at the starting point of the line shows raw data captured by a sensor, and an icon at the end point shows a final visualization method. Icons on the line named mPATH module

show analysis methods of data.

Users can interactively register and delete icons from the interface, and connect and disconnect icons by mouse operation. When the command executed successfully, the visual on the interface also changes immediately. A user can access detail of each icon by clicking right button. It provides menu to access internal parameter, and a visualization window itself.

mPATH Core

mPATH Core is a main part of the system, that manages modules which are currently utilized. It holds a list of active modules, and provides their icons to the visual programming interface with connection information between modules. It also accepts commands to change status of modules from the visual programming interface. When the core receives a command to connect modules, it registers one module to the other mutually.

Module Manager

In the Module Manager, several modules are registered. When we utilize a module, the manager copies it and registered to the mPATH Core. This manager also provides a list of available modules to the visual programming interface.

mPATH Module

mPATH module consists of two parts: a common part and an original part for each module. The common part of the module has lists of modules per inputs and outputs of the module, which contain the addresses of connected modules.

The original part of each module contains following four methods. Developers can develop original modules by creating an object extending the object of the common part, and filling these methods.

- **getBehaviorElement(GeoShape area):** This method was called when analysis result is required by a lower module. In the method, this module must return all analyzed result in the given area. If this module has data inside, it is recommended to utilize regional-index such as R-Tree[15] to select data inside the given area efficiently. If this module works simply as a filter, it calls the same method of the upper module, and return results after processing certain analysis method.

- **afterConnectFrom(MPathModule fromModule):** This method is called when other module is connected to input. A connected module is noticed as an argument. If this module caches data to return, it is needed to load upper data and store processed data.

- **afterDisconnectFrom(MPathModule fromModule):** This method is called when a connected upper module is disconnected.

- **preNotification(MPathModule fromModule):** This method is called when the state of upper module is changed. In this method, this module should analyze again its internal data if it has data which depend on the upper module.

4 Demonstration

To demonstrate functionality of the mPATH Framework, we developed several mPATH modules, and showed results of visualization. We classified mPATH modules into input, filter, and output modules. Input modules have only output ports, and output modules have only input ports. Filters have both, and output processed data acquired from the input ports. Although output modules could flexibly output their results, we currently focus on visualization, thus all the output modules below was designed to be a viewer of behavior data.

4.1 Input Modules

GPS Movement Module

We implemented a mPATH Module of movement data acquired by a handy GPS. With this module, we can input movement data captured by Garmin eTrex[16] from local files. We can also input real time movement data from a GPS receiver of NMEA-0183 style through RS-232C port.

Digital Photo Module

We implemented a mPATH module of digital photo. This module deals with jpeg files as behavior histories of "taking pictures" by reading time stamps and location in EXIF[17] area. In addition, by matching timestamps of photos with movement data, it estimates location where the photos were taken without location information.

Vector Map Module

We implemented a module of map data in vector format. There are several popular map format for GIS like Shape format[18] for ArcGIS[19] and the format of the Geographical Survey Institute of Japan. These map contains points of station, lines of road, polygons of buildings, and so on. Users can infer a detail of behavior with information of real world by using map data as well as behavior data in the system, and also use the map as a background of visualization.

4.2 Filter Modules

Time Filter

Time filter extracts behavior histories within a specified term.

Speed Filter

Speed filter filters data by its speed. It is useful especially to infer transportation from movement data.

Waypoint Filter

Waypoint filter classifies GPS point data into movements and stops. Users can change threshold of time to detect the user's stop.

Matching Filter

Matching filter joins two input data according to location. When we input one from the map data and the other from waypoints data, this filter adds name of places to each waypoint.

Count Filter

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

4.3 Output Modules

Normal Map Viewer

To visualize spatial aspect of behavior history, we developed normal map viewer. In this viewer, every input is ordered by the geographic coordinates, so that generic map-like visualization is realized. This viewer accepts multiple data and overlays them. Figure 3 shows an example of a normal map viewer.

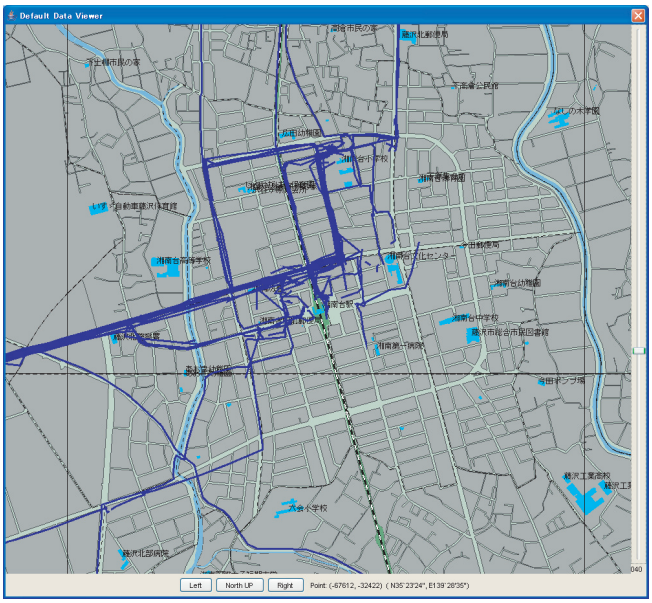

Fig. 3. Normal Map Viewer

Table Viewer

We developed table viewer to show details of data by characters and figures. This viewer lists every input data, so that it is useful especially for developing analysis method. Figure 4 shows an example of the viewer.

Weight Map Viewer

Weight Map Viewer shows weights of each geographical region visually in addition to normal map. There are several ways to visualize weights of regions such as colors of a map, scale of a map, or distortion of a map. In current implementation, we use scale of each regions to show weights. Figure 5 shows an example of a weight map viewer, which is used to emphasize the places where user passed by or took pictures.

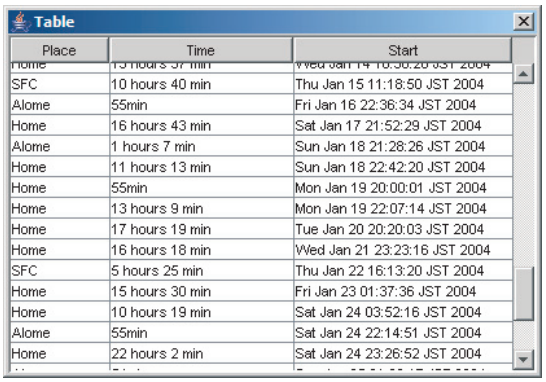

Fig. 4. Table Viewer

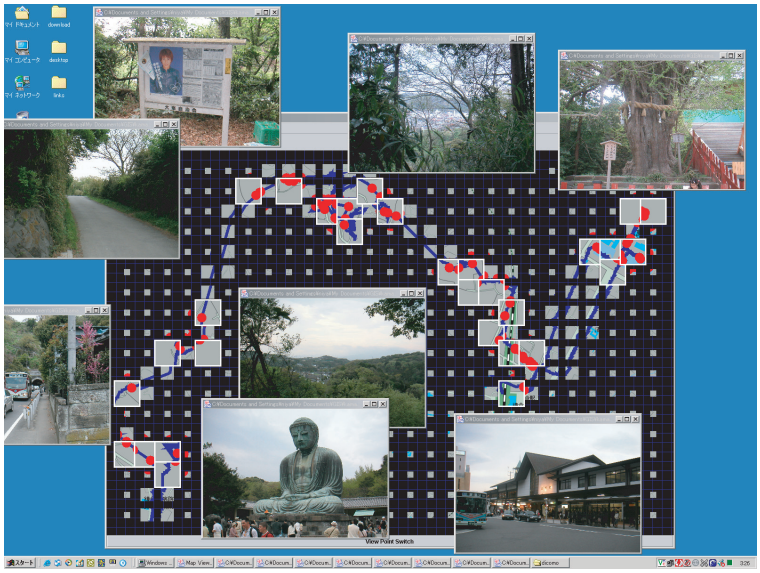

Fig. 5. Weight Map Viewer

4.4 Examples of Visualization

Using above modules, we demonstrated following visualizations.

Visualization of Behavior History on a Map

We visualized behavior history on a map. It shows not only movement data as lines, but also waypoints where he or she stayed for more than certain time. Figure 6 shows the program. We applied a time filter and a waypoint filter to GPS movement data and displayed them with vector map on a normal map viewer. Changing parameters of the time filter enables interactive observation of GPS data within a specific term.

Listing Waypoints with Name of Places

We output a list of name of places where a user stayed. We utilized matching filter with waypoint filter, and showed on a table viewer. Figure 7 shows the program, and Figure 4 is a

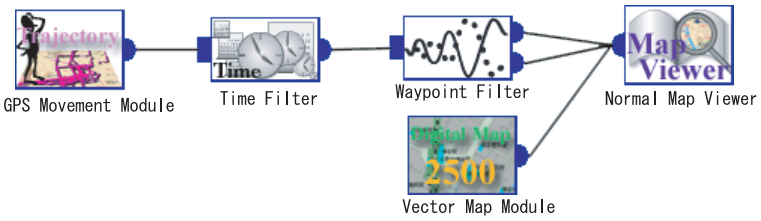

Fig. 6. A Program for Visualization Behavior History on a Map

result. We can find places where a user visited frequently. In this example, user's home and a university are listed.

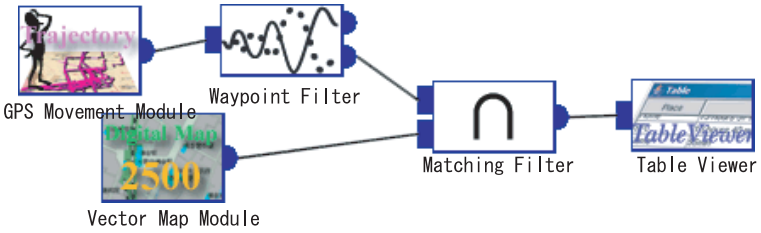

Fig. 7. A Program for Listing Waypoints

Intuitive Visualization of Past Activity

By combining count filters and a weight map viewer, we demonstrated visualization which intuitively shows places which a user interested in. We counted the time a user passed through, stay and took picture by each region. Then visualized on a weight map viewer to show a number of each region. Figure 8 shows the program of this visualization, and Figure 5 shows the result.

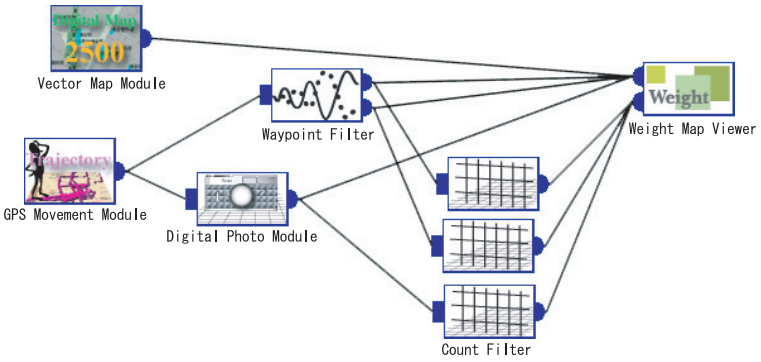

Fig. 8. A Program of Weight Visualization of Travel Log

5 Applications

Various visualization of behavior is useful to create applications for personal communication. We developed following applications on the mPATH framework. When we developed these

applications, we implemented specific functions to the application also as mPATH modules, and create applications by combining new and existing filters.

5.1 Sharing Personal Experiences

We developed a publication system of personal experiences on the web using the mPATH framework. We developed a mPATH module with web server, from which user can acquire output map by HTTP. By using the filter with an existing weblog tool, we can easily publish a personal diary with a map on which movement, photo and other behavior are shown. Figure 9 shows the screen shot of the application.

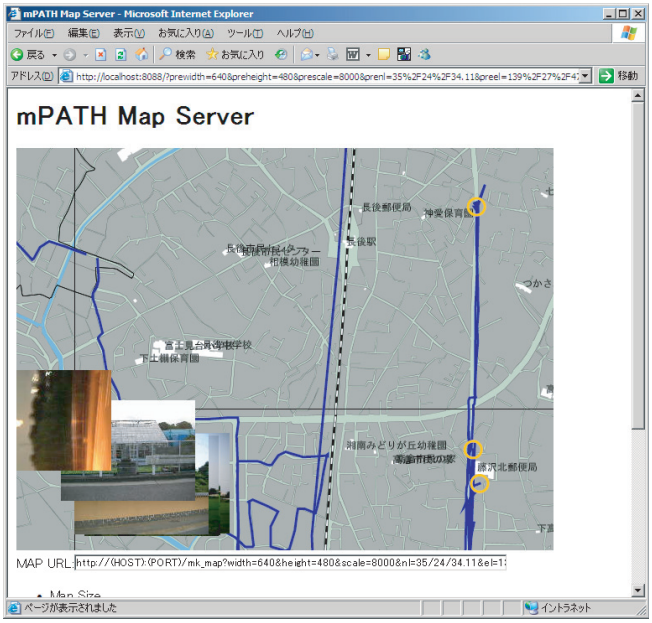

Fig. 9. Publishing Personal Experiences

5.2 mPATH View

mPATH View is an interactive viewer of personal behavior history. Using the mPATH view, we can enjoy looking arbitrary data on the 3D map. We demonstrated the viewer on the Third International Conference on Pervasive Computing[20]. mPATH View utilize a tablet PC equipped with an acceleration sensor and a magnetic sensor. For the tablet PC, we developed a 3D map viewer as a mPATH filter. The 3D map on the viewer keeps changing to synchronize with our orientation and angle, which assist our intuitive navigation of the map.

6 Evaluation

In this section we evaluate the performance of the mPATH system. It shows if the response of the system is enough to develop analysis and visualization methods interactively. It can also be used to find bottleneck of the system. We measured the performance on a machine shown in table 1 and utilized data shown in table 2. These data are collected from the experiment of Ito, who has been carrying handy GPS since June 2003.

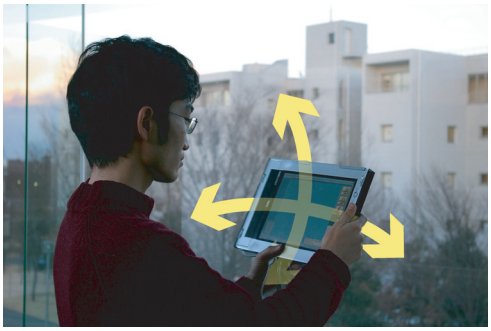

Fig. 10. Usage of the mPATH View

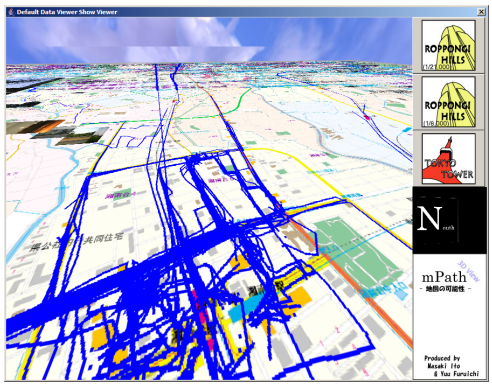

Fig. 11. Screen Shot

Table 1. Evaluation Environment

CPU	Pentium4 2.53GHz
memory	1024MB
OS	Linux 2.4.22
JDK	J2SDK 1.4.2_03

6.1 Overhead of the Component-based Architecture

We measured the overhead of the component-based architecture, since the architecture seemed to be an overhead comparing with a hard-coding implementation. We measured simple visualization method with several time filters, where we changed the number of the time filters connected.

Figure 12 shows the result of the measurement. When we increased time filter from zero to 15, the growth of the time is small. This result shows that the overhead of component-based architecture is small.

6.2 Performance of the Visualization

We measured the performance of visualization on the mPATH framework. We developed simple visualization in which movement data and map data are rendered. Then measured the time of rendering. Figure 13 shows the result. It also shows the result of rendering time only of a movement or a 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.

Table 2. Data for Evaluation

Type	GPS Movement Data	Map Data
description	Jun. 2003 – Nov. 2003	Fujisawa city
size	1,956,381 byte	9,261,000 byte

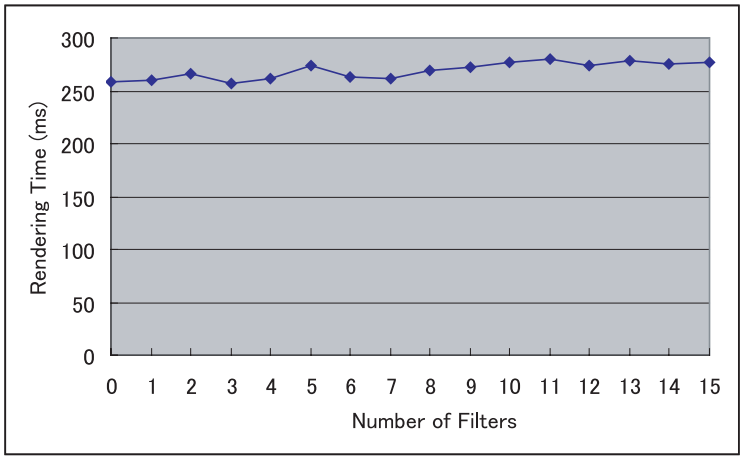

Fig. 12. Measurement of the Overhead

7 Related Works

We introduce several researches and products as related work of mPATH framework in points of view of an analysis of behavior history and a visual programming language.

7.1 Analyzing Behavior History

There are already many researches which aim to acquire and utilize behavior history. PEPYS[13] is one of the earliest fruits of such researches on the ubiquitous computing research area. PEPYS uses Active Badge[21] to acquire location information of each users, and generate text-based behavior logs like diaries.

Cyberguide[4] also accumulate location history to inform tourist attractions which a traveler prefers. This system uses GPS to acquire location information, however the system uses names of tourist attractions which already prepared with geographic coordination.

Analyzing location information described as geographic coordination is tough problem since the cost of calculation and creating users' activity model increases. [22] and [23] is trying to infer user's behavior and transportation from movement data of GPS and creating user's activity model to predict future behavior.

In the research area of the transportation planning, many researchers are trying to use GPS receivers to acquire a movement of a crowd instead of counting passers in several places of a city. They argue mainly on the accuracy and correction of raw GPS data[24][25].

7.2 Visual Programming

We introduce several visual programming system which provides users with programmability for data analysis. These systems do not focus on behavior history, but provide flexibility of data analysis and representation by visual language manner.

Many data flow visual languages are proposed[14] for music[26], image processing, scientific visualization[27][28] and creating user interface. As business applications, data mining application such as Intelligent Miner of IBM, Enterprise Miner[29] of SAS and Clementine[30] of SPSS provide visual programming language to develop analysis method.

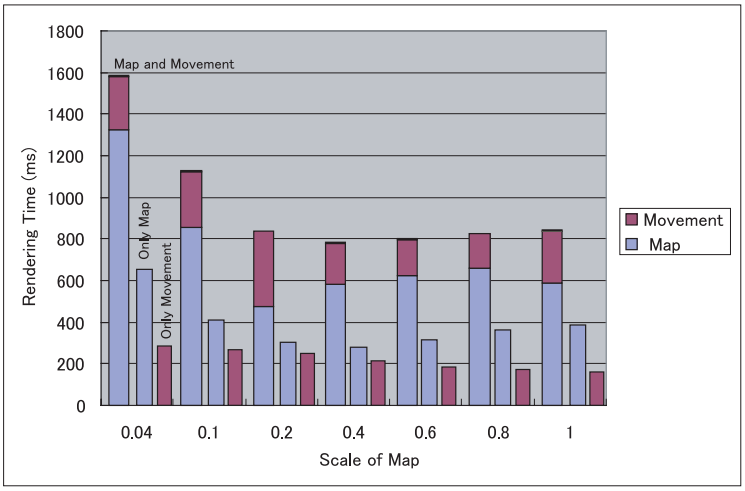

Fig. 13. Application Performance

8 Conclusion

In this paper, we presented an interactive analysis and visualization framework for behavior 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 analysis result for various applications. We implemented mPATH framework with Java language, and demonstrate constructing various viewer applications. We also evaluated performance of the framework, which shows that its performance is enough for interactive analysis.

References

1. Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, pages 263–270, Seattle, Washington, USA, August 1999. ACM.
2. Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A Survey on Sensor Networks. *IEEE Communications Magazine*, pages 102–114, August 2002.
3. Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, November 2000.
4. G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: A mobile context-aware tour guide, 1997.
5. Keith Cheverst, Nigel Davies, Keith Mitchell, and Adrian Friday. Experiences of developing and deploying a context-aware tourist guide: the GUIDE project. In *Proc. of the Sixth Annual ACM International Conference on Mobile Computing and Networking (MobiCom2000)*, pages 20–31, 2000.
6. Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The Cricket Location-Support System. In *Proc. of the Seventh Annual ACM International Conference on Mobile Computing and Networking (MobiCom2001)*, pages 32–43, August 2000.
7. Changhao Jiang and Peter Steenkiste. A Hybrid Location Model with a Computable Location Identifier for Ubiquitous Computing. In *Proceedings of the 4th international conference on Ubiquitous Computing*, pages 246–263. Springer-Verlag, 2002.

8. Daniel Salber, Anind K. Dey, and Gregory D. Abowd. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *CHI*, pages 434–441, 1999.
9. Manuel Roman, Christopher K. Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, and Klara Nahrstedt. Gaia: A Middleware Infrastructure to Enable Active Spaces. In *IEEE Pervasive Computing*, pages 74–83, 2002.
10. Gregory D. Abowd. Software Engineering Issues for Ubiquitous Computing. In *proceedings of ICSE'99*, pages 13–19, April 1999.
11. Michael G. Lamming and William M. Newman. Activity-based Information Retrieval: Technology in Support of Personal Memory. *Information Processing*, III:68–81, 1992.
12. Jim Gemmell, Gordon Bell, Roger Lueder, Steven Drucker, and Curtis Wong. MyLifeBits: Fulfilling the Memex Vision. In *ACM Multimedia '02*, pages 235–238, 2002.
13. William M. Newman, Margery A. Eldridge, and Michael G. Lamming. PEPYS: Generating Autobiographies by Automatic Tracking. In *Proceedings of ECSCW '91*, pages 175–188, September 1991.
14. Daniel D. Hils. Visual Languages and Computing Survey: Data Flow Visual Programming Languages. *Journal of Visual Languages and Computing*, 3(1):69–101, March 1992.
15. Antonin Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of ACM SIGMOD Conference on Management of Data*, pages 47–57, 1984.
16. Garmin Ltd. Garmin eTrex Legend, 2001. `http://www.garmin.com/products/etrexLegend/`.
17. JEIDA. *Digital Still Camera Image File Format Standard (Exchangeable image file format for Digital Still Cameras: Exif) Version 2.1.* 1998.
18. Environmental Systems Research Institute Inc. *ESRI Shapefile Technical Description.* An ESRI White Paper. July 1998.
19. Environmental Systems Research Institute Inc. ArcGIS. `http://www.esri.com/software/arcgis/index.html`.
20. Masaki Ito, Yuu Furuichi, Jin Nakazawa, and Hideyuki Tokuda. mPATH View: An Interactive Behavior History Viewer for Enhancing Communication. In *Adjunct Proceedings of the Third International Conference on Pervasive Computing*, pages 93–96, May 2005.
21. R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. In *ACM Transactions on Information Systems, vol 10*, pages 91–102, January 1992.
22. Donald J. Patterson, Lin Liao, Dieter Fox, and Henry Kautz. Inferring High-Level Behavior from Low-Level Sensors. In *Proceedings of The Fifth International Conference on Ubiquitous Computing (UBICOMP2003)*, pages 73–89, 2003.
23. Daniel Ashbrook and Thad Starner. Learning Significant Locations and Predicting User Movement with GPS. In *Sixth International Symposium on Wearable Computers(ISWC 2002)*, pages 101–108, October 2002.
24. Jean Wolf, Randall Guensler, and William Bachman. Elimination of the travel diary: An experiment to derive trip purpose from GPS travel data. *Notes from Transportation Research Board, 80th annual meeting*, January 2001.
25. Peter Stopher, Philip Bullock, and Qingjian Jiang. GPS, GIS and personal travel surveys: an exercise in visualisation. *25th Australasian Transport Research Forum Incorporating the BTRE Transport Policy Colloquium*, October 2002.
26. Cycling'74. Max/MSP. `http://www.cycling74.com/products/maxmsp.html`.
27. Sait Dogru, Vijay Rajan, Keith Rieck, James R. Slagle, Bosco S. Tjan, and Yewei Wang. A Graphical Data Flow Language for Retrieval, Analysis, and Visualization of a Scientific Database. *Journal of Visual Languages & Computing*, 7(3):247–265, 1996.
28. Advanced Visual Systems. `http://www.avs.com`.
29. SAS Institute Inc. SAS Enterprise Miner. `http://www.sas.com/technologies/analytics/datamining/miner/`.
30. SPSS Inc. SPSS Clementine. `http://www.spss.com/clementine/`.