Development of a Software Framework for a Networked Wearable Augmented Reality System

Daisuke Takada[1], Takefumi Ogawa[1,2], Kiyoshi Kiyokawa[1,2] and Haruo Takemura[1,2]

[1] Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
[2] Cybermedia Center, Osaka University
1-32 Machikaneyama, Toyonaka, Osaka 560-0043, Japan
`takada@lab.ime.cmc.osaka-u.ac.jp`,
{`ogawa, kiyo, takemura`}`@ime.cmc.osaka-u.ac.jp`

Abstract. We have been researching a software framework to facilitate the development of a networked wearable augmented reality (AR) system. We developed the simple toolkit to provide three fundamental functionalities: database management, communication, and rendering. We then improved the communication functionality further and developed a dynamic priority control technique for filtering AR annotations. This technique is able to give higher priority to the more necessary annotations according to a user's position and the angular velocity of the viewing direction.

1 Introduction

Augmented reality (AR) is a form of virtual reality that supplements the real world with virtual information rather than creating an entirely synthetic environment [1, 2]. Among its variety of applications, a wearable AR system has been actively studied for a variety of purposes, including navigation in a real environment [5, 6, 14–17]. To share annotation data among users and to allow information providers to update annotation data from time to time, a networked wearable AR system that follows a client-server model is useful.

When huge amounts of annotation information are managed in a large real environment, the system must pay the cost of picking up the information needed by each user and then displaying the information in the correct locations. Wearable computers with low power will not perform such tasks. Furthermore, in a multi-user environment, the consistency of the data stored in all users' wearable computers must be guaranteed, and this task also comes at a cost. Therefore, an effective networked AR system is one which manages annotations in a central server, treats each wearable computer as a client and distributes annotations to these clients.

Many studies have been performed to find a technique for efficient distribution of the annotations stored in the server and shared with multiple clients[7–9, 12, 13, 11]. First, some techniques use the idea of an AOI (Area of Interest), which is defined as a fixed space around the user for filtering annotations to be transferred [4, 9, 10, 13]. In particular, Han et al. integrate multiple users' AOIs which overlap each other to respond to the increase of clients [9]. Also, some techniques are being researched to evaluate the interest of the user for each virtual object and to transfer information about the virtual objects which have been prioritized to be of higher interest. Park et al. proposed a technique which evaluates the interest of each user and creates a separate set of all users for each virtual object and evaluates the sending priority using the interest and position of each virtual object [13].

Although some studies have used a networked wearable AR system, some problems unique to a networked wearable AR system must be solved to enable users to walk around a large area and provide them with the appropriate information according to user's position. For example, a technique is needed which transfers annotation information to clients with consideration of the unstable and narrow network environments used by wearable PCs–such as the PHS (Personal HandyPhone System).

We developed a simple toolkit to provide three functionalities: database management, communication and rendering. We then improved the communication functionality further and developed a dynamic priority control technique for filtering annotations. This technique is able to give higher priority to more necessary annotations according to a user's position and the angular velocity of the viewing direction. In the following, the software development toolkit and the dynamic priority control technique are explained in order.

2 Software Development Toolkit for Networked Wearable AR Systems

In this section, we describe our software development toolkit for building a networked wearable AR system. Section 2.1 describes the typical process flow of a networked wearable AR system using a client-server model. Section 2.2 discusses the main functionalities for the toolkit along with particular consideration of the location-based and real-time nature of the wearable AR system.

2.1 Process flow of a networked wearable AR system

Figure 1 shows a typical flow diagram of a networked wearable AR system using a standard client-server model. The server maintains a location-based annotation database and controls data transmission among the clients. A client, a wearable AR system with wireless networking, shows annotation data about the real environment to the user. The main process of the server waits for requests for connection from the clients. When the process accepts a connection request, it establishes a duplex connection and creates a new process to handle the client (Figure 1 (I)). We assume each client knows network information of the server in the area in advance. However, a general scheme to find a server from the client's location must be developed in the future for automatic connection establishment.

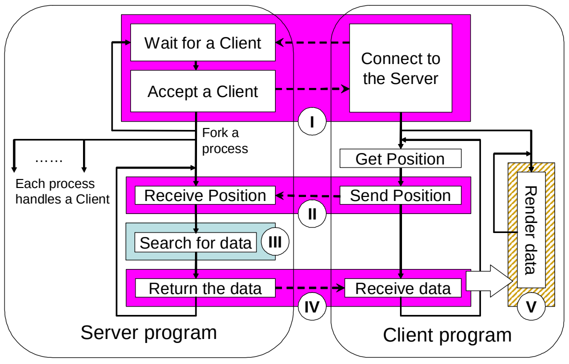

Fig. 1. The process flow of a networked wearable AR system

Once the connection is established, the position and orientation information of the client is periodically sent to the server as the client moves around the real environment (Figure 1 (II)). The server then searches for appropriate annotation data based on the location information (Figure 1 (III)) and sends the data back to the client (Figure 1 (IV)). Finally, the client interprets and renders the received annotation data in the user's view (Figure 1 (V)).

2.2 Functionality requirements of the toolkit

As shown in Section 2.1, a development toolkit for networked wearable AR systems is expected to support three types of functions: database management, communication and rendering. In the following, we explain each of three functionalities of our software development toolkit.

Database library: The database library allows a user program (mainly, the server system) to query the annotation database by position and orientation. The library supports several features designed for the wearable AR system for fast and effective data transfer. An item of the annotation database is composed of an area ID, position, orientation, and one or multiple data entities (text, image, sound, movie clip, polygon, etc.). For quick searching and scalability, an entire area managed by a database is partitioned into sub-areas hierarchically. Based on the location information, the library efficiently searches for only the data in the client's current area or its neighborhoods, and sorts the results in order of likelihood of their visibility from the user's viewpoint, which is calculated from the relative distance, apparent size and position, and pre-defined significance of each annotation. Currently the database is built using a plain text file. However, for compatibility and maintainability, it is better to build on a SQL-based standard database system and support export/import functions in XML format.

Communication library: The communication library handles all network issues between the server and clients. Annotation datasets retrieved from the database are transmitted to the corresponding clients through sender and receiver functions of the library. Transmission priority among a set of data which was originally set by the database library is recalculated in real time. Each data item is also transmitted using a level-of-detail scheme. That is, a simple text label is first sent and then other large content (e.g., images and polygons) follows in a progressive fashion. By monitoring the current available bandwidth periodically, the sender and receiver functions dynamically increase and decrease the data size that is sent out at any time. If there is room for additional data, the sender function voluntarily retrieves and sends out data of neighboring areas (prefetch). On the other hand, data transmission of each entity is stopped and resumed in response to decreased and increased priority, respectively. The receiver function stores all entries of the received data in a local cache to prevent redundant data transfer.

Rendering library: The rendering library is responsible for the final representation of the annotation data displayed in the user's view. It interprets the raw data and renders them appropriately in consideration of the user's preferences and graphics power. For example, a client with high graphics power will render a very detailed polygon model, while another with low graphics power will render the same data in coarser detail. Each client is able to override many properties of the visual attributes of the annotation data, such as the font size and layout policies of the text labels. In future, some of the visualization techniques should also be supported [3]. A dedicated layout engine, similar to an HTML browser, must also be provided that displays combinations of the data elements proportionally.

3 Hierarchical Data Structure Considering a Real Environment

This section describes the method of managing annotations at the server. The areas covered by the server are subdivided into small areas hierarchically and each area corresponds with a space in the real environment. Annotations belong to one of these areas. Figure 2 shows a hierarchical structure of areas and annotations.

Each area has information about the real space corresponding with the area. According to the definition of an area, a category of the area (e.g., "Room," "Floor," "Building," "Area," "Town," "City," "State/Prefecture") can be set. These categories are used to find annotation information.

Each user can select a yes-or-no policy for each category. Eight different policies can be used:

1. When the user is in the area
   (a) include annotations in the area into the search result

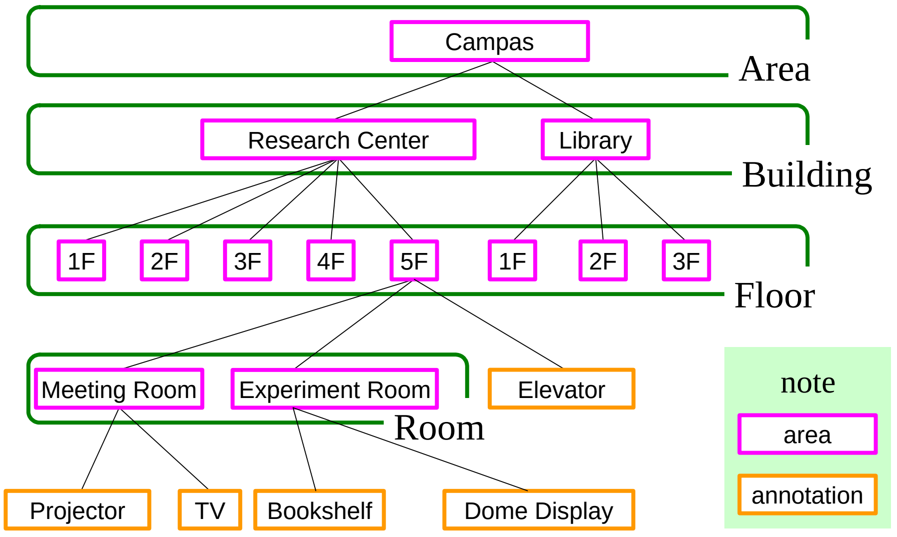

Fig. 2. Hierarchical data structure containing areas and annotations

   (b) traverse each child area for a search
   (c) traverse the parent area for a search
2. When the area is traversed from a child area
   (a) include annotations in the area into the search result
   (b) traverse each child area for a search
   (c) traverse the parent area for a search
3. When the area is traversed from the parent area
   (a) include annotations in the area into the search result
   (b) traverse each child area for a search

For example, when a user is in a room and needs annotations which are only in that room and in the buildings around the area, he can set "yes" for policies 1(a), 1(c), and 2(c) for categories "Room" and "Floor"; "yes" for 2(a), 2(c), 3(a) for "Building"; "yes" for 2(b) for "Area" to restrict the target area to search annotations.

4 Dynamic Priority Control Technique

4.1 Proposed method

As the result of the search, the system can get a set of annotations which satisfy the conditions. After that, the system evaluates a transfer priority for each annotation and sends them according to this priority. Figure 3 shows the position of user A and annotation O. In this case, the priority of O for the user A, $p(O, A)$ is defined in this expression:

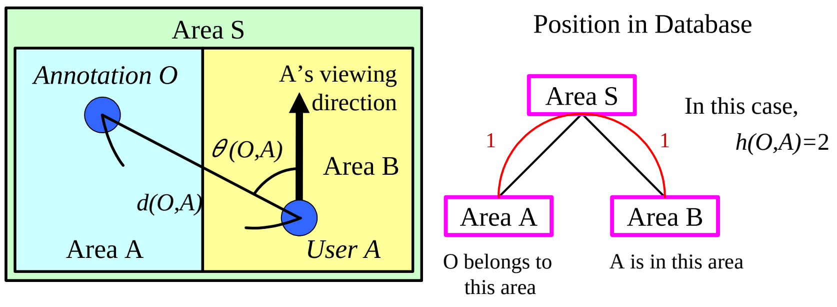

Fig. 3. Parameters used for the priority evaluation

$$p(O,A) = (1 - \frac{h(O,A)}{H(A)})\{\alpha(1 - \frac{d(O,A)}{D(A)}) + (1-\alpha)(1 - \frac{\theta(O,A)}{\pi(A)})\} \quad (1)$$

α is a constant defined as $0 < \alpha < 1$. When α is small, annotations which are nearer to the user's viewing direction are prioritized, and when α is big, annotations which are nearer to the user are prioritized. And when the distance between the area where the user is and the one where the annotation belongs is further down in the tree structure, the priority becomes low because the area in this research should correspond to the semantic space in a real environment. Also, $D(A), \pi(A), H(A)$ are constants used to normalize variables $d(O,A), \theta(O,A), h(O,A)$, and are given by the user when the connection is established.

When the user's viewing direction is moving fast and is big, it is assumed that the user himself is looking around. And when it is stable, it is assumed that the user needs the information which is in the same direction. In this case, the priority depending on the user's behavior can be evaluated by relating the movement of the user's viewing direction with α. In this case, equation (1) is expanded as below:

$$p(O,A) = (1 - \frac{h(O,A)}{H(A)})\{s(1 - \frac{d(O,A)}{D(A)}) + t(1 - \frac{\theta(O,A)}{\pi(A)})\} \quad (2)$$
$$s = \frac{S(A)w(A)}{S(A)w(A) + \{(1 - S(A)\}\{(W(A) - w(A)\}}$$
$$t = \frac{(1 - S(A))(W(A) - w(A))}{S(A)w(A) + \{(1 - S(A)\}\{(W(A) - w(A)\}}$$

$w(A)$ is the angular velocity which expresses the movement of the user's viewing direction. When $w(A)$ is big, annotations which are nearer to the user's viewing direction are prioritized, and when $w(A)$ is small, annotations which are nearer to the user are prioritized by equation (2). $W(A)$ is a constant for normalizing $w(A)$ and is given by the user when the connection is established, as with the other constants.

It might not be required to determine the wait for $d(O, A)$ and $\theta(O, A)$ automatically using the angular velocity. In this case, there is a constant $S(A)$ in equation (2). $S(A)$ is defined as $0 < S(A) < 1$ and is given by the user when the connection is established, as with other the constants.

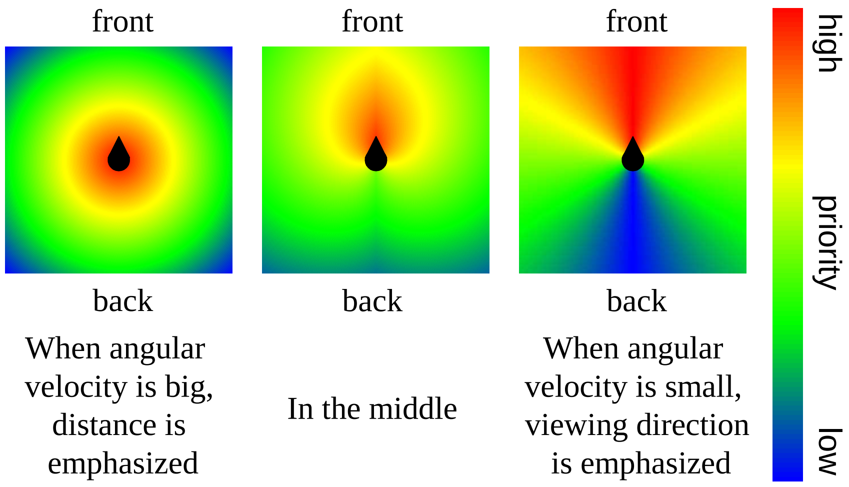
front
front
front
back
back
back
When angular velocity is big, distance is emphasized
In the middle
When angular velocity is small, viewing direction is emphasized
high
priority
low

Fig. 4. Difference of priority by moving the user's viewing direction

Figure 4 shows the change of the priority while the user's viewing direction is moving. When the angular velocity is big, annotations which are nearer to the user are prioritized and when the angular velocity is small, ones which are nearer to the user's viewing direction are prioritized. And when the angular velocity is in the middle, some of the annotations of each are prioritized.

4.2 Experiment

Figure 5 shows the simulation environment. The user follows the green line and looks around two times left and right at 90 degrees when the user arrives at some specified points. The velocity of the user is 4 km/h. We measured the angular velocity while the user was looking around in a real environment and used this value as the angular velocity of the user while he is looking around in the simulation. In this experiment, all of the annotations are the same size in the 3D space, then nearer annotations are rendered bigger in the user's view. Then, we defined the value $satisfy(A, t)$, where t indicates the time and A is the user.

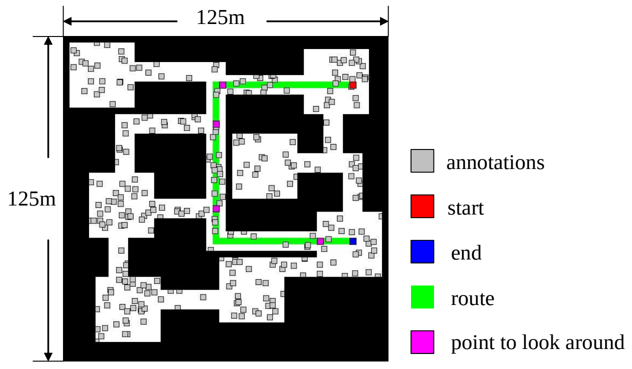

Fig. 5. Settings for simulation experiment

$$satisfy(A,t) = \frac{\sum_{O \in viewable(t)} \frac{1}{d(O,A)}}{\sum_{O \in should\_be\_viewable(t)} \frac{1}{d(O,A)}} \qquad (3)$$

The format of the image data used in this experiment is progressive jpeg; thus, 20% of the entire data is enough to be roughly recognized. Then, $viewable(t)$ is defined as a set of annotations for the 20% of the entire image data that was transferred. And $should\_be\_viewable(t)$ is defined as a set of annotations which should be in the user's view at time t.

Figure 6 shows the change of $satisfy(O, A)$ with time for three priority evaluation techniques; distance based, angle based, and our proposed technique. We used $D(A) = 130$, $\pi(A) = 190$, $W(A) = 110$, $S(A) = 0.5$ as constants in the proposed method. As shown in Figure 6, the proposed method gives higher satisfaction than other methods.

5 Conclusion

First, we reported our past and ongoing studies on a networked wearable augmented reality system. We developed a simple toolkit to provide three functionalities: database management, communication and rendering. Also, we developed a dynamic priority control technique for filtering annotations. This technique is able to give higher priority to more necessary annotations according to the user's position and angular velocity of viewing direction. In future, we would like to confirm the effectiveness of our method in a real environment.

References

1. R. T. Azuma, "A survey of Augmented Reality," MIT Press Presence, Vol.4, No.6, pp.355–385, 1997.

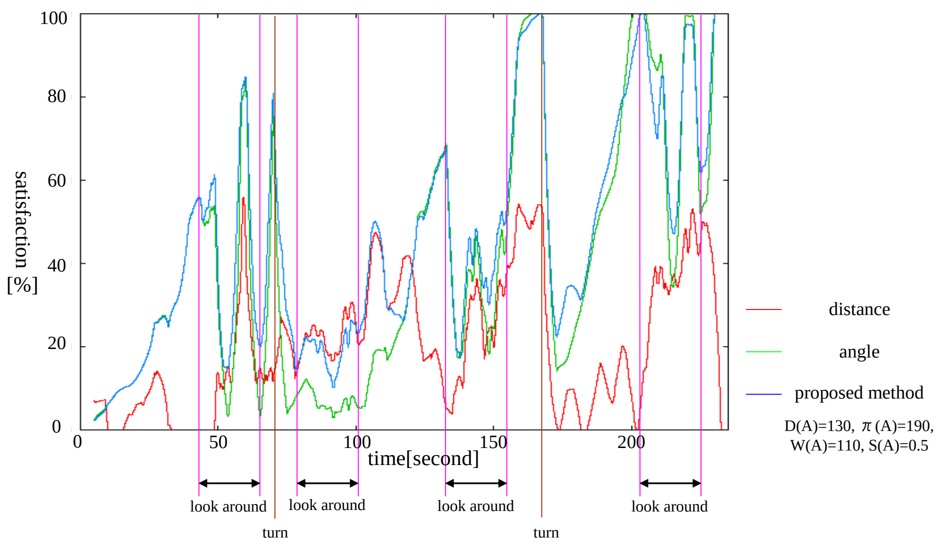

Fig. 6. The change of the satisfaction with time

2. R. T. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent Advances in Augmented Reality," IEEE Computer Graphics and Applications, Vol.21, No.6, pp.34–47, 2001.
3. B. Bell, S. Feiner, and T.Höllerer, "View Management for Virtual and Augmented Reality," Proc. of the 14th Annual ACM Symposium on User Interface Software and Technology (UIST'01), pp.101–110, 2001.
4. D. Ding and M. Zhu, "A Model of Dynamic Interest Management: Interaction Analysis in Collaborative Virtual Environment," Proc. of the ACM Symposium on Virtual Reality Software and Technology (VRST'03), pp.223–230, 2003.
5. S. Feiner, B. MacIntyre, and T. Höllerer, "A Touring Machine: Prototyping 3D Mobile Augmented Reality System for Exploring the Urban Environment," Proc. of the 1st International Symposium on Wearable Computers (ISWC'97), pp.208–217, 1997.
6. S. Feiner, B. MacIntyre, and D. Seligmann, "Knowledge-based Augmented Reality," Communication of ACM, Vol.36, No.7, pp.52–62, 1993.
7. T. A. Funkhouser, "RING: A Client-Server System for Multi-User Virtual Environments," Proc. of Symposium on Interactive 3D Graphics, pp.85–92, 1995.
8. C. Greenhalgh and S. Benford, "MASSIVE: a collaborative virtual environment for teleconferencing," ACM Transactions on Computer-Human Interaction, Vol.2, Issue.3, pp.239–261, 1995.
9. S. Han, M. Lim, D. Lee, "Scalable Interest Management Using Interest Group Based Filtering For Large Networked Virtual Environments," Proc. of the ACM Symposium on Virtual Reality Software and Technology (VRST'00), pp.103–108, 2000.
10. M. Hosseini, S. Pettifer, and N. D. Georganas, "Visibility-based Interest Management in Collaborative Virtual Environments," 2004.

11. M. R. Macdonia, M. J. Zyda, D. R. Pratt, P. T. Barham, and S. Zeswitz, "NPSNET: A Network Software Architecture For Large Scale Virtual Environments," MIT Press Presence, Vol.3, No.4, pp.265-287, 1994.
12. B. Ng, A. Si, R. W. H. Lau, and F. W. B. Li, "A Multi-Server Architecture for Distributed Virtual Walkthrough," Proc. of the ACM Symposium on Virtual Reality Software and Technology (VRST'02), pp.163–170, 2002.
13. S. Park, D. Lee, M. Lim, and C. Yu, "Scalable Data Management Using User-Based Caching and Prefetching in Distributed Virtual Environments," Proc. of the ACM Symposium on Virtual Reality Software and Technology (VRST'01), pp.121–126, 2001.
14. R. Tenmoku, M. Kanbara, and N. Yokoya, "A Wearable Augmented Reality System Using Positioning Infrastructures and a Pedometer," Proc. of 7th IEEE International Symposium on Wearable Computers (ISWC'03), pp.110–117, 2003.
15. B. Thomas, B. Close, J. Donoghue, J. Squires, P.D. Bondi, M. Morris, and W. Piekarski "ARQuake: An Outdoor/Indoor Augmented Reality First Person Application," Proc. of 4th IEEE International Symposium on Wearable Computers (ISWC'00), pp.139–146, 2000.
16. B. Thomas, W. Piekarski, D. Hepworth, B. Gunther, and V. Demczuk, "A Wearable Computer System with Augmented Reality to Support Terrestrial Navigation," Proc. of 2nd International Symposium on Wearable Computers (ISWC'98), pp.168–171, 1998.
17. V. Vlahakis, J. Karigiannis, M. Tsotros, N. Ioannidis, and D. Stricker, "Personalized Augmented Reality Touring of Archaeological Sites with Wearable and Mobile Computers," Proc. of 6th International Symposium on Wearable Computers (ISWC'02), pp.15–22, 2002.