EGVE Symposium (2008)
B. Mohler and R. van Liere (Editors)

A View Management Method for Mobile Mixed Reality Systems

Fumihisa Shibata[1], Hiroyuki Nakamoto[1 †], Ryoichi Sasaki[1 ‡], Asako Kimura[2,3], and Hideyuki Tamura[1]

[1]Graduate School of Science and Engineering, Ritsumeikan University
[2]PRESTO, Japan Science and Technology Agency
[3]Research Organization of Science and Engineering, Ritsumeikan University

Abstract

This paper describes a view management method for annotations on mobile mixed reality systems. In recent years, mobile devices, such as mobile phones and PDAs, have rapidly gained popularity. Thus, MR systems using mobile devices are expected to be a rich field. However, the mobile devices' components (screen and application executable memory) are very small compared to those of notebook computers. Also, the processor does not have enough computational power. Thus, it is difficult for such devices to apply existing view management techniques. We propose a view management method that consists of seven sub-functions. Each sub-function is implemented by a simple algorithm, while a system developer assembles required sub-functions into a customized view management component. We implemented some applications based on the proposed method and discussed the effectiveness of the view management algorithm.

Categories and Subject Descriptors (according to ACM CCS): H.5.1 [Multimedia Information Systems]: Artificial, augmented, and virtual realities

1. Introduction

Mixed Reality (MR) is an emerging technology to merge the real world and the virtual world seamlessly in real time. Early in its development, MR systems were developed on high performance computers. Recently, mobile devices, such as mobile phones and PDAs, have rapidly gained popularity using the same type of technology used in MR. Thus, MR systems using mobile devices are expected to be a logical next step and a rich field for research and development [WS03, PW03, MLB04, GFO06, SW07]. One of the most remarkable characteristics of mobile devices is that they can be used anytime and anywhere. Mobile devices provide us an easy way to merge electronic data into the real world.

There are any challenges still to be solved to realize effective mobile MR systems. View management is one of the most important issues for developing practical MR systems. View management methods for MR systems have been studied in various respects.

Leykin proposed an automatic algorithm that determines positions of annotations by analyzing background textures [LT04]. The most important problem of view management is how to decide the spatial layout of the annotations in the view plane. In previous work, Bell et al. [BFH01] proposed a view management method for augmented and virtual reality applications. Their method adjusts the position and size of the controllable objects at every frame to satisfy the visibility constraints of the controllable objects. Azuma et al. [AF03] proposed a view management algorithm based on identifying existing clusters. However, most previous work focused on where and how to arrange the annotations, on the assumption that the screen size was large and the CPU power was sufficient to execute complex algorithms. From the viewpoint of intuitive presentation, Tenmoku's work focused on how to emphasize the user-viewed objects and the corresponding annotation to display links between annota-

† Current Affiliation: Toshiba Corporation
‡ Current Affiliation: Mitsubishi Electric Corporation

tions and real objects clearly [TKY06]. How to emphasize the annotations or to make the annotations conspicuous is also an important issue in view management.

From the viewpoint of portability, manufacturers try to minimize the physical size and maximize the battery life of mobile devices. As a result, the mobile devices' display screen and application executable memory are very small compared to those of notebook computers. In addition, the clock speed of the CPU installed in mobile devices is very slow. On the other hand, small physical size enables the user to vary the position and orientation of the mobile device easily. In other words, the freedom in viewing the position of mobile devices is greater than that of head-mounted displays. Thus, it is necessary to consider these platform-specific characteristics when designing MR systems that make use of mobile devices.

View management for mobile devices should be discussed from a different standpoint. Considering that mobile devices have a small screen, not many annotations can be glanced at the same time. Therefore, a complex algorithm that optimizes the spatial layout of the annotations is not useful. The small screen requires an easy user interface that enables the user to explore neighboring annotations rather than a complete label placement algorithm that arranges all annotations. A label placement strategy may depend on what the MR application does and where the MR application is used. Therefore, a view management method should have a mechanism that can deal with a variety of scenes. In addition, it is necessary to consider how to present annotations clearly on a small screen.

In this paper, we propose a view management method for annotations that consists of seven sub-functions. Each sub-function is implemented by a simple algorithm. Dividing into sub-functions enables a system developer to assemble required sub-functions into a customized view management component.

The remainder of this paper is organized as follows. In section 2 we discuss related work. Section 3 describes the proposed view management method in detail. Section 4 and 5 show the experimental results and applications that use the proposed method. Finally, a conclusion and future work are provided in Section 6.

2. Related Work

The challenge in managing labels is one of traditional issues in computer science. This problem is called as 'Label Placement', 'Map-Labeling', or 'Name Placement.' Some of earliest research addressed on how to place a large number of labels on a geographic map. As Azuma mentioned [AF03], there are a lot of works concerning map labeling [Wol]. Contrary to what we deal with, they can spend a lot of time calculating the placement, because a target of label placement is a static map. However, the time available is not infinite. Any complete search will be impractical because the search space grows exponentially [CMS95]. Therefore some researchers proposed approaches that employ various heuristics. This kind of approach is called 'greedy approach.' One of greedy approaches is to place each object to the best location locally available in some order such as priority [BFH01]. Simulated annealing is also available for label placement [KGV83]. However, we did not adopt these complex algorithms because they require too much computational power to execute. Complex algorithms decrease frame rate and low frame rate spoils usability.

3. View Management

3.1. Overview

Typically, mobile devices have the following features:

- Small and low resolution display
- Limited application execution memory
- Processor running at a slower speed

Considering these features, we expect that the view management method for mobile devices should satisfy the following conditions:

- It adopts a simple algorithm that does not require much memory, rather than a complex algorithm that wastes too much memory and computational power.
- It is equipped with a mechanism to cope with a small screen.

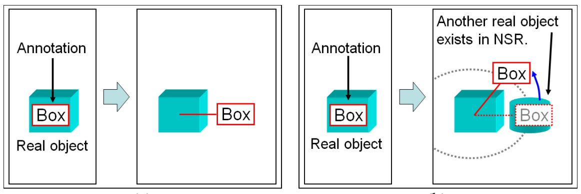
We propose a modularized method for view management that meets the above conditions. Our proposed method consists of seven sub-functions. Each sub-function executes a simple placement rule. An actual view management component is built of some sub-function modules. In Azuma's work [AF03], he compared four different placement algorithms in an AR application. The compared algorithms were greedy algorithm, gradient descent, adaptive simulated annealing, and his novel algorithm based on identifying existing clusters. From the viewpoint of computation time analysis, greedy algorithm proved to be the fastest of all the algorithms. Therefore, we use greedy approaches to make each sub-function. Of course, greedy approaches have a vulnerability to being trapped inside local minima. However, we think it is a rare case that our method falls within local minimum, because not many annotations are displayed on the screen. In the following subsection, we describe the algorithm of each sub-function.

3.2. Sub-functions for View Management

3.2.1. Layout of Annotations

This section describes rearranging methods of annotations. The following description shows three cases that require rearranging annotations and their typical algorithms.

(1) Occluding real objects
The simplest algorithm of annotating a real object using MR techniques is drawing annotations on the real object. However, this algorithm is not user-friendly in some cases. For example, for an operation support system, annotations should not cover real objects which are handled by the user. In such a situation, annotations should be replaced so as not to overlay the real objects. Our system arranges the annotation around the real objects and draws a line between the annotation and the object. Figure 1 (a) illustrates an example of this situation. In addition, it is conceivable that the replaced annotation occludes other important real objects. Some algorithms can prevent such cases [BFH01]. Here, we classify the real scene into two regions: "superimposable regions" (SR) and "non-superimposable regions" (NSR). If an annotation is in NSR, the annotation is replaced in the neighboring SR. Figure 1 (b) illustrates this case. The searching for the neighboring SR starts from just right (or left) of the real object and it follows counterclockwise (or clockwise) on the oval. An application developer can determine the start point and search direction.



(a) (b)

Figure 1: *These figures illustrate view management for occluding real objects. Figure (a) shows the case that there is no other real object around the target object. Figure (b) shows that another important object exists beside the target object.*

(2) Mutual overlap of annotations
In the case where multiple annotations are presented at the same time, mutual overlap of annotations may sometimes occur. Azuma et al. gave some solutions to this problem [AF03]. However, when the application does not show so many annotations, the rearranging method using priority value of annotation is easily feasible and does not need so much CPU power. Figure 2 shows the process of rearrangement that considers priority values of annotations. Using priority value of annotation, some algorithms can be conceived. For example, the annotation whose priority is low is shown translucently behind the high priority annotation or is moved to prevent mutual overlap. Our method adopts the algorithm that moves annotations of low priority.

(3) Getting out of the frame
When the annotation gets partially occluded by the edge of the frame, we should consider how to present the annotation in the entirety. One of solutions is that such annotations should be moved to the place where the user can see. In this case, the system flips the position of the annotation horizontally. Another solution is that the system removes such annotations from the screen. Figure 3 illustrates these view management ideas. Our system adopts both algorithms. Thus, a system developer can select the preferable algorithm that fits to the concerned MR application. In both cases, these algorithms are applied when the half of the annotation is out of the frame.

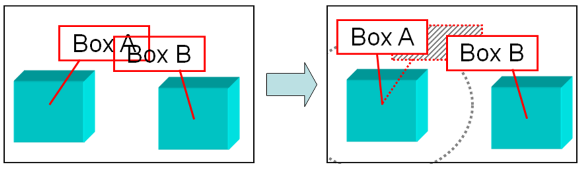
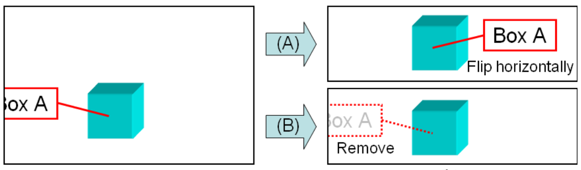


Figure 2: *When mutual overlap of annotations occurs, the annotation which has low priority is relocated to another position. This figure shows the case that the annotation of Box B has higher priority value than that of Box A.*



(a) (b)

Figure 3: *When the half of the annotation is out of the frame, the system flips the position of the annotation horizontally (right upper) or the system removes the annotation (right lower).*

3.2.2. Styles of Annotations

This section describes annotation styles that should be changed according to the situations. These styles aim to emphasize annotations.

(4) Size of annotations
We have some choices of presenting methods of annotations. One of them is to present annotations just like perspective projection. By applying this presentation method, the user can intuitively understand the depth perception of annotations and real objects in depth direction(Figure 4 (a)). However, the user cannot read the annotation when they stand away from the annotation. On the other hand, the opposite method is to present all annotations in the same size. However, this method spoils the user's perception of the relative depth of annotations (Figure 4 (b)). One of our approaches takes a moderate

stand between the former two methods. The distance between the user and the annotation is classified into several levels. The system adjusts the size of the annotations according to the levels. The relationship between the distance and the level depends on the type of MR application. We call this method 'semi-perspective' method.
We also propose another approach to control the size of the annotations. The second approach varies the size according to the position of the annotations. We call this method 'fish-eye' method. The magnification is calculated by the following function $f(x)$:

$$f(x) = p_1 \frac{(p_2+1)(L-x)L}{p_2(L-x)+L} + p_3$$

where p_1, p_2, p_3 are parameters to control the rate of variation. L is the half-length of the diagonal line of the screen and x represents the length between the center of the screen and the position of the annotation. Figure 5 illustrates an image of this method.

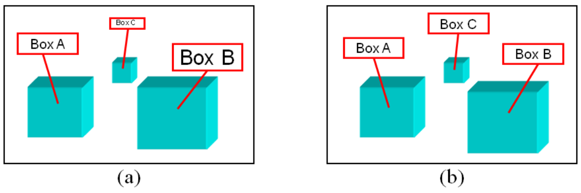


Figure 4: *Figure (a) illustrates the image of the perspective method. The size of the annotations varies with the depth of the objects. In this figure, the letters of Box C are too small to read clearly. Figure (b) shows the image of opposite method, which displays all annotations in the same size. The user cannot perceive the depth of the objects from the size of the annotations.*

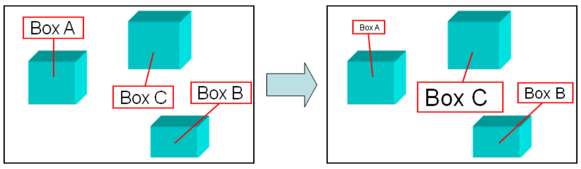


Figure 5: *The size of annotations is varied according to their positions. The right figure illustrates an image of magnified annotations.*

(5) Color of annotations
The color of annotations influences their visibility as much as their size [LT04]. The annotations should contrast strikingly with the background color. Figure 6 shows a conceptual image. In the figure, frames of the annotations are painted with the color which is opposite to the color of the background. We adopt 'hue circle' as a guideline and pick up six basic colors from it. The system distinguishes the background color into the six colors and selects its opposite color as the edge color of the annotation. This operation relatively takes much computational time.

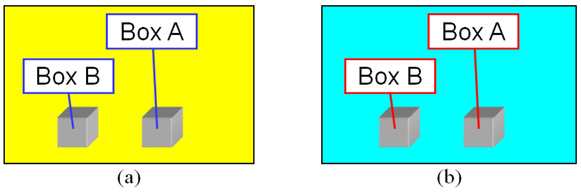


Figure 6: *Figure (a) shows the case that the color of the background is red. The frames' color of the annotations is blue. On the other hand, the frames are colored with red since the color of the background is blue.*

(6) Transparency of annotations
The transparency of annotations can be varied according to a kind of criterion. For example, Uratani et al. [UMKT05] proposed a technique which varies the transparency of the annotations as a function of relative depth. We propose another technique which makes the annotations conspicuous. Our technique varies the transparency of the annotations in proportion to their distance from the center of the screen. The transparency is calculated by the following function $g(x)$:

$$g(x) = \frac{x}{L}$$

where L is the half-length of the diagonal line of the screen and x represents the length between the center of the screen and the position of the annotation. Figure 7 shows a conceptual image.

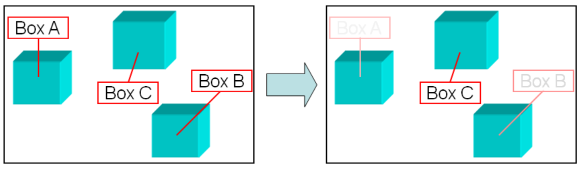


Figure 7: *The transparency of annotations is varied according to their positions. The right figure illustrates an image of transparent annotations.*

3.2.3. Target of Annotations

This section describes a technique which emphasizes the target of annotations. Our technique makes the annotations conspicuous.

(7) Highlighting real objects
A user occasionally does not understand a significant point on the screen. In this situation, the significant real object is highlighted in order that the user notices

it [TKY06]. The system draws a rectangle around the target real object. Figure 8 illustrates this function.

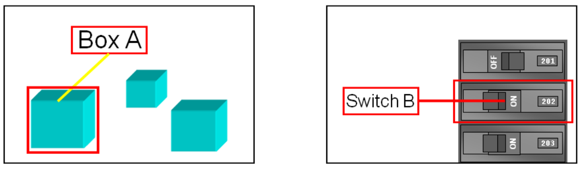


Figure 8: *The target of the annotation is enclosed with red lines to highlight the real object.*

4. Experimental Results

We implemented our view management methods to both a notebook PC and a PDA. The notebook PC used here was a Dell XPS M1210 with a 2.0GHz Intel Core 2 Duo CPU, 2.0GB RAM and a GeForce GO 7400 GPU. It was a video see-through system equipped with a Logitec Qcam Fusion QVX-13 USB 2.0 camera. The resolution of captured images was VGA size. For the notebook PC, we used the ARToolKitPlus [WS07] as a tracking mechanism. An HP hx4700 with a 624MHz Intel XScale PXA270 CPU and a camera device, Life View Fly-CAM-CF (QVGA), was used as an experimental PDA. In addition, the ARToolKit [KB99, KBP*00] was used as a tracking mechanism. Both the notebook PC and PDA were implemented as a heavy-load client (HLC) [SHF*06].

Figure 9, 10, 11 show the screenshots using sub-functions (1), (2), and (6) respectively.

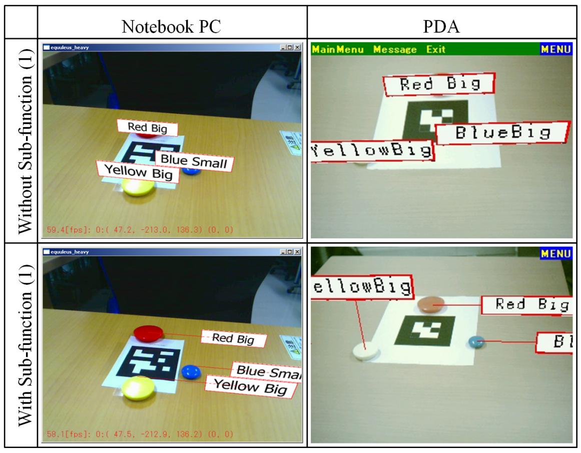


Figure 9: *These images are the results of sub-function (1) for the notebook PC and PDA. The upper images show the results without the sub-function, and the lower images show the results with the sub-function.*

We measured computational time of each sub-function in

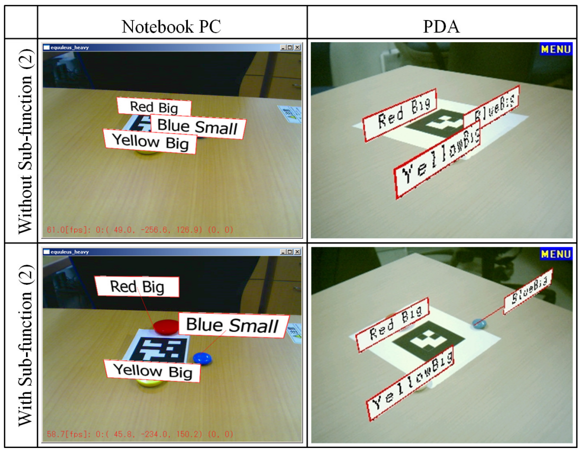


Figure 10: *These images are the results of sub-function (2) for the notebook PC and PDA. The upper images show the results without the sub-function, and the lower images show the results with the sub-function.*

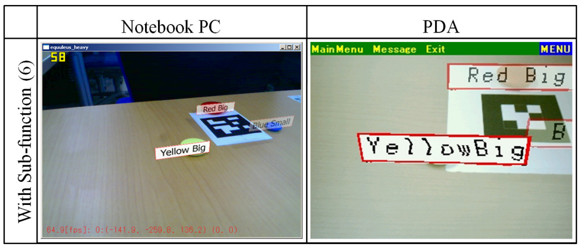


Figure 11: *These images are the results of sub-function (6) for the notebook PC and PDA.*

section 3.2. This time does not include rendering time. Table 1 summarizes the computation time measurements of the sub-functions. Sub-function (4-a) and (4-b) correspond to 'semi-perspective' method and 'fish-eye' method respectively. We also measured computation time for capturing images.

Of course, if a developer combines multiple sub-functions for his/her application, computation time will increase. However, the total time is the sum of the computation time of each sub-function. The sub-function (5) for PDAs takes much time compared with other sub-functions. This is because the sub-function (5) requires too much memory access to check colors of the captured image. Currently, this measurement shows the following two matters: For notebook PCs, these sub-functions are appropriate as view management method. For PDAs, sub-functions except sub-function (5) are applicable as view management method. In the near future, memory access speed will increase due to rapid development of GPU. The sub-function (5) will become available in that condition.

Mobile device	(1)	(2)	(3)	(4-a)	(4-b)	(5)	(6)	(7)	Capture
Notebook PC	0.003	0.006	0.162	0.003	0.002	5.669	0.002	0.002	3.854
PDA	15.78	15.67	15.71	0.04	0.11	670.84	0.05	0.33	166.39

Table 1: *This table shows computation time in milliseconds. Each value is an average of 100 frames.*

5. Applications

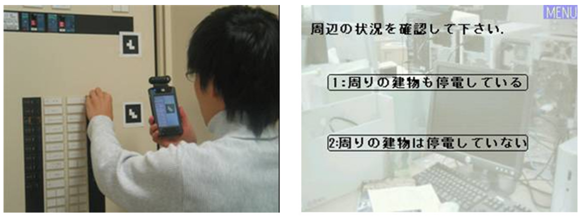
We have developed three applications using our view management method. Each application adopts a suitable combination of the functions, which we propose.

5.1. Inspection Support System for Switchboard

5.1.1. Overview

The purpose of this system is to show operating procedures for inspecting a switchboard. Annotations are overlaid on toggle switches of the switchboard panel using MR techniques. In particular, this application supports the user who is not an expert in electricity check for leakage. We suppose that this system is useful for inspection when experts can not come to the job site because of traffic impassability after a disaster such as an earthquake. The user checks the switchboard as shown in Figure 12 (a). The user dialogically answers the question about environmental conditions. Figure 12 (b) illustrates an example of a dialogue.

In this application, annotations are overlaid near the real toggle switches. This application includes view management sub-functions (1), (2), (3), and (6), because there are many circuit breakers, and non-specialists operate the application.


周辺の状況を確認して下さい.

1:周りの建物も停電している

2:周りの建物は停電していない

(a) (b)

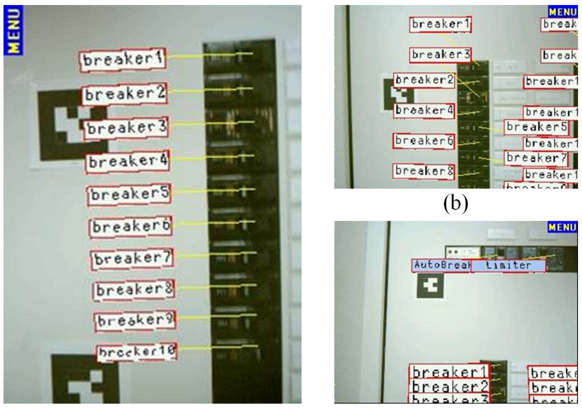
Figure 12: *The user checks the switchboard using a PDA. Figure (b) shows an example of a dialogue (in Japanese Kanji). It directs the user to look around at his/her surroundings.*

The markers were attached on the switchboard and each marker was a square whose side was 5 cm. We experimented using a real switchboard in our laboratory.

5.1.2. Results and Discussion

Figure 13 shows example images of the application. We decided NSR over each toggle switch region on the switchboard panel beforehand (see Figure 12 (a)). Annotations were rearranged on the left side of these switch areas (see Figure 13 (a)). This indicates that view management sub-function (1) is successful.

In this application, because the user does not need to perceive relative depth of annotations, annotations are presented to the user at the same size irrespective of the distance between the camera and the switchboard. However, in this condition, the annotations were sometimes mutually overlapping. Then, view management sub-function (2) was applied to prevent mutual overlaps of annotations (see Figure 13 (b)). In this figure, mutual overlaps of annotations do not occur. However, a yellow line connecting the annotation and each toggle switch is not understandable because of the line intersections. As this is not user-friendly, we have to solve this problem. Figure 13 (c) shows an example it without a sub-function (2).



(a) (b) (c)

Figure 13: *We arranged NSR over each toggle switch on the switchboard so that the user can see the switches while operating them (Figure (a)). Figure (b) shows an example with sub-function (2) and Figure (c) shows an example without sub-function (2).*

5.2. Recovery Support System of Network Server

5.2.1. Overview

The purpose of this application is to assist the user in inspection and recovery of network servers after a blackout. Network servers consist of Web Server, Mail Server, and DNS Server. An uninterruptible power supply (UPS) is adopted with each network server. In this application, the following four modes are included.

(A) UPS Checking Mode
For recovering the network server, a UPS is needed to maintain the power supply of the server when a blackout occurs. This mode supports a check of whether the UPS is on or off. If the power is off, a message prompts the power switch to be turned on. Then, the remaining battery charge of the UPS is superimposed on the screen. View management sub-functions (1), (2), and (6) are built in this mode.

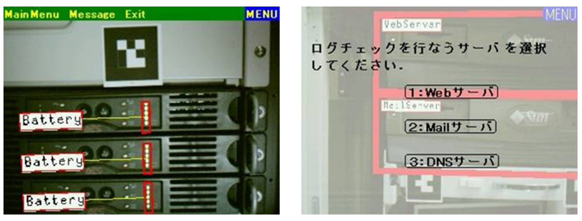
(B) Activation of the Power Supply
After checking the UPS, the application explains how to start the servers. This application targets three kinds of servers: Web Server, Mail Server, and DNS Server. The user checks whether these servers can start normally.

(C) Log Checking Mode
If something is wrong with these servers, this mode furnishes a log check function to a user who has expertise in maintaining the server.

(D) UPS Manual Service
This mode informs a user on each LED's status on the UPS. The system stores the information like a conventional manual. Therefore, the user can check the position of each LED by superimposed annotations and can understand the meaning of each LED status by the conventional manual. The view management sub-function (7) is used to indicate the position of the LEDs.

5.2.2. Results and Discussion

In this application, the same PDA as mentioned in Section 3.1 was used. In mode (A), the UPS information was shown using various sub-functions of the view management techniques. For example, in the case of sub-function (1), we configured regions of the server and the UPS to SRs, and regions of the LED indicators on the UPS to NSRs. Figure 14 (a) shows an example image that includes highlighting indicators of significant points on the screen. This is realized by sub-function (7). Moreover, Figure 14 (b) shows an example image of "Log Checking Mode." the user can read the instruction and recognize the servers at the same time since the instruction displayed on the foreground is translucent.



ログチェックを行なうサーバを選択してください.

1:Webサーバ
2:Mailサーバ
3:DNSサーバ

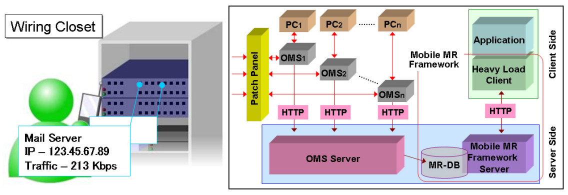
(a) (b)

Figure 14: *Red boxes are displayed on the screen to highlight the LED indicators on the UPS (Figure (a). The user can see both the instruction and the real environment simultaneously.*

5.3. LAN Wiring Support System

5.3.1. Overview

This system supports LAN wiring that takes much time and effort because of the complexity of the structured cabling. Figure 15 (a) is the conceptual image of the application. At the wiring closet, the users can see superimposed annotations, which include IP addresses, MAC addresses, traffic, and so on. Figure 15 shows the block diagram of the system. We developed the system based on the framework, which we proposed [SHF*06]. The Open Micro Server (OMS) is used for gathering network information. The OMS is a Linux-based small computer that has three Ethernet connectors. It works as a network bridge and simultaneously it collects information such as IP addresses, MAC addresses, protocol type, traffic, and so on. We put the OMSs into LAN to monitor network packets and they notify network information to the MR content server via the OMS server at regular intervals. Therefore, the users can check current network information to find out a problem in the LAN.



(a) Concept Image (b) Block Diagram

Figure 15: *This figure is the conceptual image of LAN Wiring Support System.*

5.3.2. Results and Discussion

Figure 16 shows the screenshot of LAN Wiring Support System. View management sub-functions (1), and (2) are implemented on this application. The system relocated annotations to avoid overlapping them with the connectors to which a cable is connected. We have also tried the application with sub-function (3). However, for the PDA, sub-function (3) is inapplicable because the size of the annotations is large in comparison with the size of the screen. On the other hand, it is applicable for the PC, which has a large screen. Our approach is reasonable because each sub-function can be enabled according to mobile devices in use.

6. Conclusion and Future Work

This paper proposes a view management method to be adopted for low-performance mobile devices such as PDAs. The view management method is classified into seven sub-functions, and functions are selected properly for each application. We have also implemented the view management sub-functions into three applications. In addition, we discussed the effectiveness of each sub-function.

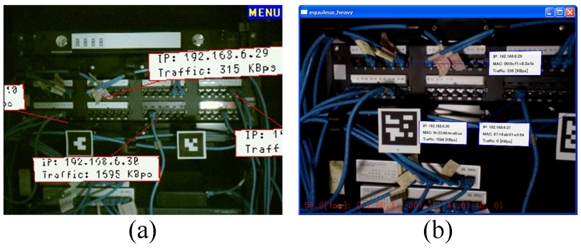


Figure 16: *Figure (a) is the screenshot of the PDA without sub-function (3). Figure (b) is the screenshot of the PC with sub-function (3).*

In the future, we have to add a new sub-function for the view management method. We consider a new sub-function that rearranges annotations without any crossing of linking lines between annotations and real objects. In Azuma's work [AF03], some algorithms were introduced. However, we have to propose another algorithm that can be realized by the PDA. Moreover, we will think of view management that is appropriate for virtual objects such as 3D models.

Acknowledgement

This research was partly supported by a Grant-in-Aid for Scientific Research (B) of the Ministry of Education, Culture, Sports, Science and Technology, No.17200039, 2005. We thank Ryuhei Tenmoku, Yuji Hayashi, Hiroyuki Iwakura, Kazuya Tanaka, and Shotaro Arakawa for their support. Thanks for Christopher Stapleton for editing the English version of this paper.

References

[AF03] AZUMA R., FURMANSKI C.: Evaluating label placement for augmented reality view management. In *Proc. 2nd IEEE and ACM Int'l Symp. on Mixed and Augmented Reality* (2003), pp. 66–75.

[BFH01] BELL B., FEINER S., HÖLLERER T.: View management for virtual and augmented reality. In *Proc. Symp. on User Interface Software and Technology* (2001), pp. 101–110.

[CMS95] CHRISTENSEN J., MARKS J., SHIEBER S.: An empirical study of algorithms for point-feature label placement. *ACM Trans. on Graphics 14*, 3 (1995), 203–232.

[GFO06] GÜVEN S., FEINER S., ODA O.: Mobile augmented reality interaction techniques for authoring situated media on-site. In *Proc. 5th IEEE and ACM Int'l Symp. on Mixed and Augmented Reality* (2006), pp. 235–236.

[KB99] KATO H., BILLINGHURST M.: Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *Proc. 2nd Int'l Workshop on Augmented Reality* (1999), pp. 85–94.

[KBP*00] KATO H., BILLINGHURST M., POUPYREV I., IMAMOTO K., TACHIBANA K.: Virtual object manipulation on a table-top AR environment. In *Proc. Int'l Symp. on Augmented Reality* (2000), pp. 111–119.

[KGV83] KIRKPATRICK S., GELATT C. D., VECCHI M. P.: Optimization by simulated annealing. *Science 220*, 4598 (1983), 671–680.

[LT04] LEYKIN A., TUCERYAN M.: Automatic determination of text readability over textured backgrounds for augmented reality systems. In *Proc. 3rd IEEE and ACM Int'l Symp. on Mixed and Augmented Reality* (2004), pp. 224–230.

[MLB04] MÖHRING M., LESSIG C., BIMBER O.: Video see-through AR on consumer cell-phones. In *Proc. 3rd IEEE and ACM Int'l Symp. on Mixed and Augmented Reality* (2004), pp. 252–253.

[PW03] PASMAN W., WOODWARD C.: Implementation of an augmented reality system on a PDA. In *Proc. 2nd IEEE and ACM Int'l Symp. on Mixed and Augmented Reality* (2003), pp. 276–277.

[SHF*06] SHIBATA F., HASHIMOTO T., FURUNO K., KIMURA A., TAMURA H.: Scalable architecture and content description language for mobile mixed reality systems. In *Proc. 16th Int'l Conf. on Artificial Reality and Telexistence* (2006), pp. 122–131.

[SW07] SCHMALSTIEG D., WAGNER D.: Experiences with handheld augmented reality. In *Proc. 6th IEEE and ACM Int'l Symp. on Mixed and Augmented Reality* (2007), pp. 3–15.

[TKY06] TENMOKU R., KANBARA M., YOKOYA N.: A new view management method for wearable augmented reality systems -emphasizing the user-viewed object and the corresponding annotation-. In *Proc. 12th Eurographics on Virtual Environments* (2006), pp. 127–134.

[UMKT05] URATANI K., MACHIDA T., KIYOKAWA K., TAKEMURA H.: A study of depth visualization techniques for virtual annotations in augmented reality. In *Proc. IEEE Virtual Reality* (2005), pp. 295–296.

[Wol] WOLFF A.: The map-labeling bibliography. http://i11www.iti.uni-karlsruhe.de/˜awolff/map-labeling/bibliography/.

[WS03] WAGNER D., SCHMALSTIEG D.: First steps towards handheld augmented reality. In *Proc. 7th IEEE Int'l Symp. on Wearable Computers* (2003), pp. 127–135.

[WS07] WAGNER D., SCHMALSTIEG D.: ARToolKitPlus for pose tracking on mobile devices. In *Proc. 12th Computer Vision Winter Workshop 2007* (2007), pp. 139–146.