PERGAMON

Computers & Graphics 25 (2001) 799–810

COMPUTERS & GRAPHICS

www.elsevier.com/locate/cag

User interface management techniques for collaborative mobile augmented reality

Tobias Höllerer[a,*], Steven Feiner[a], Drexel Hallaway[a], Blaine Bell[a], Marco Lanzagorta[b], Dennis Brown[b], Simon Julier[b], Yohan Baillot[b], Lawrence Rosenblum[b]

[a] *Department of Computer Science, Columbia University, 1214 Amsterdam Avenue, Mail Code 0401, New York, NY 10027, USA*
[b] *Naval Research Laboratory, 4555 Overlook Avenue S.W., Washington, DC 20375, USA*

Abstract

Mobile augmented reality systems (MARS) have the potential to revolutionize the way in which information is provided to users. Virtual information can be directly integrated with the real world surrounding the mobile user, who can interact with it to display related information, to pose and resolve queries, and to collaborate with other users. However, we believe that the benefits of MARS will only be achieved if the user interface (UI) is actively managed so as to maximize the relevance and minimize the confusion of the virtual material relative to the real world. This article addresses some of the steps involved in this process, focusing on the design and layout of the mobile user's overlaid virtual environment.

The augmented view of the user's surroundings presents an interface to context-dependent operations, many of which are related to the objects in view—the augmented world is the user interface. We present three UI design techniques that are intended to make this interface as obvious and clear to the user as possible: information filtering, UI component design, and view management. *Information filtering* helps select the most relevant information to present to the user. *UI component design* determines the format in which this information should be conveyed, based on the available display resources and tracking accuracy. For example, the absence of high accuracy position tracking would favor body- or screen-stabilized components over world-stabilized ones that would need to be exactly registered with the physical objects to which they refer. *View management* attempts to ensure that the virtual objects that are displayed visually are arranged appropriately with regard to their projections on the view plane. For example, the relationships among objects should be as unambiguous as possible, and physical or virtual objects should not obstruct the user's view of more important physical or virtual objects in the scene. We illustrate these interface design techniques using our prototype collaborative, cross-site MARS environment, which is composed of mobile and non-mobile augmented reality and virtual reality systems. © 2001 Published by Elsevier Science Ltd.

Keywords: Augmented reality; Wearable computing; Information filtering; User interface design; View management

1. Introduction

Augmented reality (AR) systems integrate virtual information into a user's physical environment so that the user will perceive that information as existing in the environment. Computer graphics can be spatially registered with, and overlaid on, geographic locations and real objects to provide visual AR. Examples of potential AR applications include aircraft cockpit control [1], assistance in surgery [2], viewing hidden building infrastructure [3], maintenance and repair [4,5], and parts assembly [6,7].

We are especially interested in the user interface (UI) issues that arise when designing mobile augmented reality systems (MARS), which allow users to roam untethered, outdoors or indoors. Examples of MARS

*Corresponding author. Tel.: +1-212-939-7116; fax: +1-212-666-0140.

E-mail address: htobias@cs.columbia.edu (T. Höllerer).

0097-8493/01/$ - see front matter © 2001 Published by Elsevier Science Ltd.
PII: S0097-8493(01)00122-4

prototypes include: the *Touring Machine* [8], which shows the user information about buildings and landmarks as she navigates a university campus; *Situated Documentaries* [9], which allow the user to experience multimedia stories as part of a physically placed hypertext system; *ARQuake* [10], an outdoor/indoor AR game based on the videogame Quake; and the Battlefield Augmented Reality System, which is designed to provide situational awareness information to warfighters in an urban environment [11].

These MARS prototypes share a set of common traits:

- Each system consists of a wearable computer with 3D graphics hardware, position and orientation trackers, a see-through head-worn display, and a wireless network interface [8,11–13]. An example is shown in Fig. 1.
- Multiple MARS users are free to roam through an urban environment. Each user performs one or more tasks (such as "Follow a route between two specified points"), which can be acted upon sequentially or concurrently.
- The surrounding environment contains many physical objects whose sight and sound are essential to the performance of the users' tasks.
- The systems help users accomplish their tasks by providing them with relevant information about their environment. For example, this might include names and other properties of buildings and infrastructure that may or may not be directly visible from a user's current location.
- Users can interact with the information presented to them; for example, by creating annotations that can be attached to locations or objects. Exchanging annotations constitutes one method of collaboration between the users.
- A user may not be tracked accurately at certain times due to location and environmental conditions. Tracking inaccuracies may show up as static or dynamic errors in orientation or position readings, and can vary greatly in magnitude over time.
- A supervisory *command center* oversees the actions of the mobile users and allows stationary users to interact with the roaming users and the environment using workstation and virtual environment UIs. Command center users receive information from mobile users and can send them additional information about the environment and their tasks.
- The prototype MARS environments are represented by virtual models that contain on the order of a few dozen buildings each, and several hundred objects, such as windows, doors, and underground tunnels. (Models for real applications will need to be orders of magnitude larger.)

Fig. 1. MARS prototype.

As described above, MARS applications differ from most virtual environment applications in many ways, including the size of the physical environments that users traverse, the importance of the physical environment and how virtual information is integrated with it, the quantity and range of virtual information that can be presented to and modified by users, and the potentially large variability in tracking accuracy over time. Based on our experience developing MARS testbeds at Columbia University and NRL, we have attempted to address these issues through a set of techniques for designing MARS UIs: information filtering, UI component design, and view management.

The large amount of virtual information that can be displayed, coupled with the presence of a richly complex physical world, creates the potential for clutter. Cluttered displays can overwhelm the user with unneeded information, impacting her ability to perform her tasks effectively. We address clutter through information filtering. *Information filtering* means culling the information that can potentially be displayed by identifying and prioritizing what is relevant to a user at a given point in time. The priorities can be based on the user's tasks, goals, interests, location, or other user context or environmental factors.

While information filtering determines the subset of the available information that will be displayed, it is still necessary to determine the format in which this information is to be communicated, and how to realize that format in detail. Registration accuracy, or how accurately the projected image of a virtual object can be positioned, scaled, and oriented relative to the real world, is an important factor in choosing the right UI format. Registration accuracy is determined by tracking system accuracy, which, as the mobile user moves about, may vary for a variety of reasons that depend on the tracking technologies used. Therefore, if information is always formatted in a way that assumes highly accurate registration, that information will not be presented effectively when registration accuracy decreases. To address this issue, *UI component design* determines the format in which information should be conveyed, based on contextual information, such as the available display resources and tracking accuracy. This technique determines the concrete elements that comprise the UI and information display.

Filtering and formatting information is not enough—the information must be integrated with the user's view of the physical world. For example, suppose that a selected set of annotations is simply projected onto the user's view of the world such that each is collocated with a physical object with which it is associated. Depending on the user's location in the world (and, thus, the projection that they see), annotations might occlude or be occluded by other annotations or physical objects, or appear ambiguous because of their proximity to multiple potential referents. *View management* attempts to ensure that the displayed information is arranged appropriately with regard to its projection on the view plane; for example, virtual or physical objects should not occlude others that are more important, and relationships among objects should be as unambiguous as possible.

Fig. 2 shows these three UI management techniques as steps in a MARS UI management pipeline. Note that we do not claim that these steps form a complete UI management model. Instead, we see them as subsets of the more general design phases of content planning, UI planning, and UI realization. *Content planning* determines the information that is to be conveyed to a user using presentation goals, user models, and online databases of information and taxonomic knowledge. *UI planning* determines the best format in which to give a user access to that information, taking into account the available media, and display and interaction technologies. *UI realization* (or *content realization*)

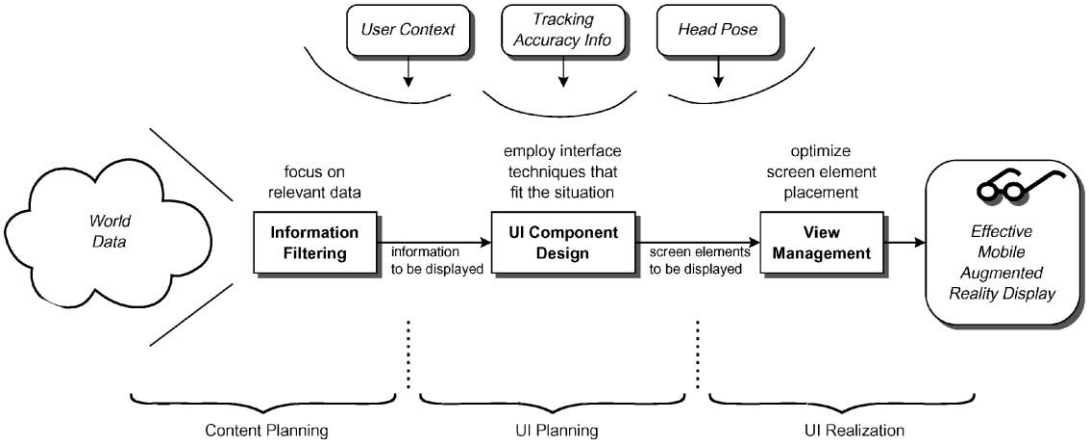

Fig. 2. Information filtering, UI component design, and view management as parts of a MARS UI management model.

finalizes concrete presentations in each of the media employed. All these techniques must be applied dynamically, since the user's tasks, the tracking accuracy, and the relative location of the user to the surrounding physical environment may change frequently.

In the following sections, we will focus on the application of these techniques in our MARS prototypes. Section 2 describes our model for information filtering. UI component design is discussed in Section 3. Section 4 presents our approach to view management. Finally, we present our conclusions and future research directions in Section 5.

2. Information filtering

Information-rich environments have the potential to overwhelm a user through the sheer volume of data that they can present. Filtering such presentations to prevent clutter and to improve human performance has long been recognized as an important technique for information display systems [14]. Information filtering culls the information that can potentially be displayed by identifying and prioritizing what will be relevant to a user at a given point in time. The filtering strategy we have developed exploits the fact that AR is a *situated user interface* [15,16] that depends on the user's location, physical context, tasks, and objectives.

2.1. Spatial model of interaction

Our information filtering approach is based in part on the *spatial model of interaction*. This model, developed by Benford and Fahlén [17], is an extremely general mechanism to determine whether two objects A and B are capable of perceiving and interacting with one another. Each object is surrounded by a *nimbus*. The *nimbus* defines the region over which the object can be perceived by other objects within a specific medium. Objects that are aware also possess a *focus*. The *focus* defines the medium-specific region over which an object is capable of perceiving and interacting with other objects. Objects are capable of interacting with one another when their foci and nimbi overlap for at least one medium.

The spatial model is well suited for the problems of information filtering. For example, it allows asymmetric interaction between two objects: if A's focus intersects with B's nimbus, but B's focus does not intersect with A's nimbus, then A can perceive and interact with B but not vice versa. Thus, an "overseer" (who possesses an extremely large focus and extremely small nimbus) could observe all other objects but not be observable by any other object. Furthermore, there are no constraints on the size and form of the focus and nimbus. They can be of arbitrary size and shape (e.g., asymmetric or disjoint) and may be discrete or continuous.

Specific examples of the model have been implemented in the MASSIVE and DIVE systems [18], which take different approaches to computing awareness. For example, in DIVE, awareness is a binary function, where A is aware of B if A's focus overlaps with B's nimbus. In contrast, in MASSIVE, foci and nimbi are scalar fields radiating from point-sized objects, focus and nimbus values are sampled at each object's position, and A's level of awareness of B is the product of B's value in A's focus and A's value in B's nimbus.

2.2. Objective and subjective properties

Our information filter has been designed to show only sufficiently important information to the user at any time. However, the importance of a piece of information depends on the user's current context (including his location and tasks). More specifically, we assume that each user is assigned a series of tasks. For each *task*, the user has to interact with a series of objects in a determined way. To model these effects, we assume that users can interact with objects through a set of media and that users and objects each possess both objective and subjective properties.

The original implementations of DIVE and MASSIVE assumed that only three media were available: audio, text, and graphics. Since we take into account interactions with both real and virtual worlds, we consider a wider range of interaction media. Since each medium has different physical properties, the medium has an impact on the importance of an object. For example, consider the two media of wireless communications and physical interaction. For a user to exchange data with a system by wireless communications, it must be within transmission range, which can be miles; in contrast, for that user to interact physically with the same system, it must be at arm's length at most. Thus, whether an object can currently participate in a task (and is important to the task) differs, depending on the medium in which the task is performed.

Objective properties are the same for all users, irrespective of the tasks they are carrying out. Such properties include the object's classification (for example whether it is a building or an underground pipe), its location, its size, and its shape. This can be extended by noting that many types of objects have an *impact zone*—an extended region over which an object has a direct physical impact. For example, wireless networks have a finite transmission range. This region might be represented as a sphere whose radius equals the maximum reliable transmission range. (A more accurate representation could take into account the antenna characteristics, and masking and multipath effects of buildings and terrain, by modeling the impact zone as a

series of interconnected volumes.) Because of their differing physical properties, the same object can have different impact zones in different media.

Subjective properties attempt to encapsulate the domain-specific knowledge of how a particular object relates to a particular task for a particular user. Therefore, they vary among users and depend on the user's task and context. We represent this data using an *importance vector*. The importance vector stores the relevance of an object with respect to a set of domain-specific and user-scenario-specific criteria. For example, in a firefighting scenario, such criteria might include whether an object is flammable or whether a street is wide enough to allow emergency vehicles to gain access. In general, the relevance is not binary-valued, but is a continuum that is normalized to the range from 0 (irrelevant) to 1 (highly relevant). For example, for the flammability criterion, the relevance might indicate the object's combustibility.

Determining the composition of the list of criteria and how a given object should be scored according to those criteria are difficult and domain-dependent tasks, which we assume will be carried out by one or more *domain experts*. For example, the sniper avoidance system described in [19] relies on US Army Training manuals that precisely codify building features and configurations.

The objective–subjective property framework can be applied to model the state of each user. Each user has their own objective properties (such as position and orientation) and subjective properties (that refer directly to the user's current tasks). Analogous to the importance vector, the *task vector* stores the relevance of a task to the user's current activities. We use a vector because the user can carry out multiple tasks simultaneously, and, by assigning weights to those tasks, different priorities can be indicated. For example, at a certain time a user might be given a task to follow a route between two points. However, the user is also concerned that she does not enter an unsafe environment. Therefore, the two tasks (route following and avoiding unsafe areas) run concurrently. The task vector is supplemented by additional ancillary information. In the route-following task, the system needs to store the way points and the final destination of the route.

2.3. Implementation

Our filtering algorithm requires the calculation of the user's focus, each object's nimbus, and the focus–nimbus interactions.

The user's focus is determined from the user's state and the medium within which a particular user–object interaction occurs. In turn, the user's state can be determined from their objective properties (including location) and their subjective properties (task vector). In our current implementation, the focus is a bounding box.

An object's nimbus is calculated as a function of the user's state, the object's state, and the medium. An object's state is defined with respect to a particular user, and depends on the object's objective properties and subjective properties. The object's subjective properties are derived from the user's state and the object's objective properties determined beforehand by a domain expert. In our approach, the nimbus is a bounding box that quantifies the importance of the object to a specific user at a specific time. This bounding box is determined by calculating the *projection* of the importance vector into the user's task vector.

Once the focus and the nimbus regions have been calculated, the level of interaction which occurs between a given focus and a nimbus is calculated. If the focus and nimbus regions do not overlap, the level of interaction is set to zero. If the user's position lies inside the nimbus, then the level of interaction is set to 1. If the focus and nimbus regions intersect, but the user's position lies outside the nimbus, the level of interaction is $1-d$, where d is the minimum distance between the nimbus's perimeter and the user's current location divided by the length of a side of the user's focus bounding box.

Fig. 4 shows the pseudocode for the main filtering loop. This algorithm is completely dynamic—it can respond to any changes to the user or to the entities in the environment. (See [19] for a more detailed explanation.) Once foci, nimbi, and their interactions have been calculated, the filtering process requires only incremental updates. Three kinds of events can trigger updates: a change in an object's state, a change in the user's tasks, and change in the user's location (greater than a threshold distance).

2.4. Filtering results

We tested the filtering algorithm in a prototype sniper-avoidance application [19]. Snipers pose a serious threat in many law enforcement, hostage rescue, and peace-keeping missions. Armed with powerful and accurate weapons, snipers exploit the 3D nature of the urban environment. Our system addresses sniper threats in two ways. First, the system provides safe routing through a urban environment avoiding sniper threats. Second, it presents information that is relevant for planning an operation to disarm a sniper.

In this domain, the user's state is determined by his position (in time and space) and the task being carried out (e.g., combat, route following, tactical planning, reconnaissance). The objects considered include buildings, cars, mine fields and snipers. Each object has objective properties and an importance vector that have been determined by careful examination of US Army manuals. For example, a sniper is *important* at all times

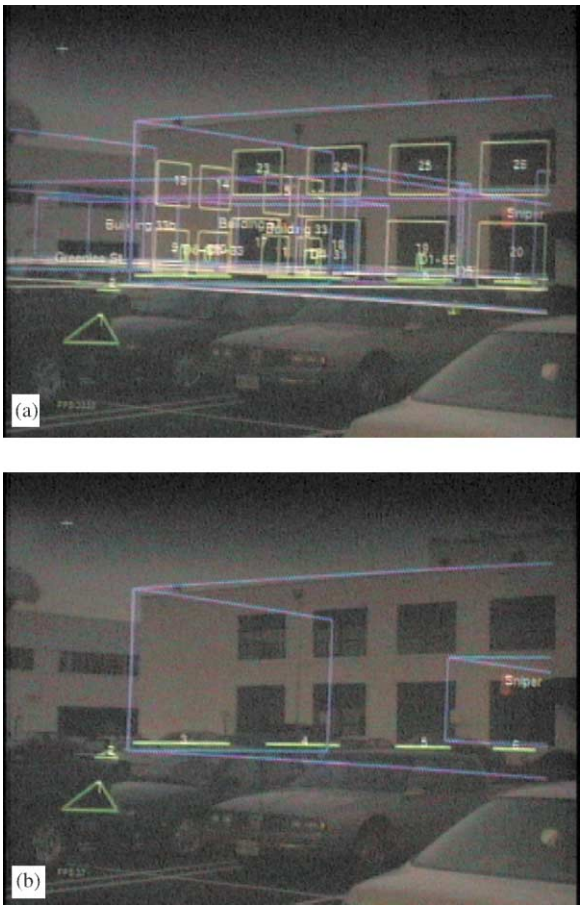

Fig. 3. Filtering in a mobile AR system (imaged through see-through head-worn display). (a) unfiltered view (b) task-oriented filtering.

and has a lethality range proportional to his weapon's range, tall buildings are *important* to prevent an ambush, and the windows of a target building are *important* in a building clearing operation.

Fig. 3 shows a pair of images captured by a camera mounted in a mannequin head that wears a see-through head-worn display. The results show the effect of the system when it is running in the *Tactical Planning* task mode. In this task mode, a user sees detailed environmental information. Fig. 3(a) shows the output from the system when filtering is disabled. The resulting image is highly cluttered; for example, data is shown about the infrastructure of buildings obscured by the currently visible building. Fig. 3(b) shows the effect of filtering, which has eliminated much of the clutter. Note that the system has not used a simple fixed-distance clipping strategy; for example, a reported sniper location in a building behind the visible building is displayed, as is part of the sniper's building infrastructure, while other closer objects are not displayed. Although we have yet to perform formal user evaluation studies, response to the filtering algorithm from prospective military users has been extremely positive. Users have commented that the algorithm eliminates superfluous information and maintains critical data that are critical to avoiding snipers.

```
Calculate user focus and nimbi for all objects with initial object
  state and user goals.
loop forever {
    if (state of an object has changed) {
        Update nimbus of that object
        Filter that object
    }
    if (user's goal has changed) {
        Update nimbi of all objects
        Filter all objects
    }
    if (user position has changed more than
       a threshold distance)
        Filter all objects
}
```

Fig. 4. Pseudocode for the main filtering loop.

In this example, our system sustains 20 frames per second in stereo. Profiling reveals that the filtering algorithm, implemented in Java on one of our mobile computers (with a 266 MHz Pentium MMX CPU) [12], completely filters an environment of 150 objects in less than 1 ms. This performance is sufficient for our current testbed.

3. UI component design

The position-tracking accuracy of a location-aware mobile system can change dynamically as a function of the user's location and other variables specific to the tracker technology used. This is especially problematic for MARS applications, which ideally require extremely precise position tracking for the user's head, but which may not always be able to achieve the necessary level of accuracy. While it is possible to ignore variable positional accuracy in an AR UI, this can make for a confusing system; for example, when accuracy is low, virtual objects that are nominally registered with real ones may be too far off to be of use.

To address this problem, we have experimented with a *UI component design* module that is responsible for adapting the system's UI automatically to accommodate changes in position tracking accuracy. Our current testbed system gracefully switches between alternative UI representations: fully registered overlay UIs, shown in Figs. 5 and 7, and a body-stabilized AR UI featuring a world in miniature (WIM) [20] (Fig. 6). This is a first step towards a more flexible automated solution, in which the system can assemble a UI from components (e.g., visually registered overlays, screen-stabilized menus, and screen or body-stabilized display and

Fig. 5. AR UI in accurate tracking mode (imaged through see-through head-worn display). Labels and features (a wireframe lab model) are registered with the physical environment.

Fig. 7. Outdoor AR UI in accurate tracking mode (imaged through see-through head-worn display). Labels and features (interactive virtual flags denoting points of interest) are registered with the physical environment.

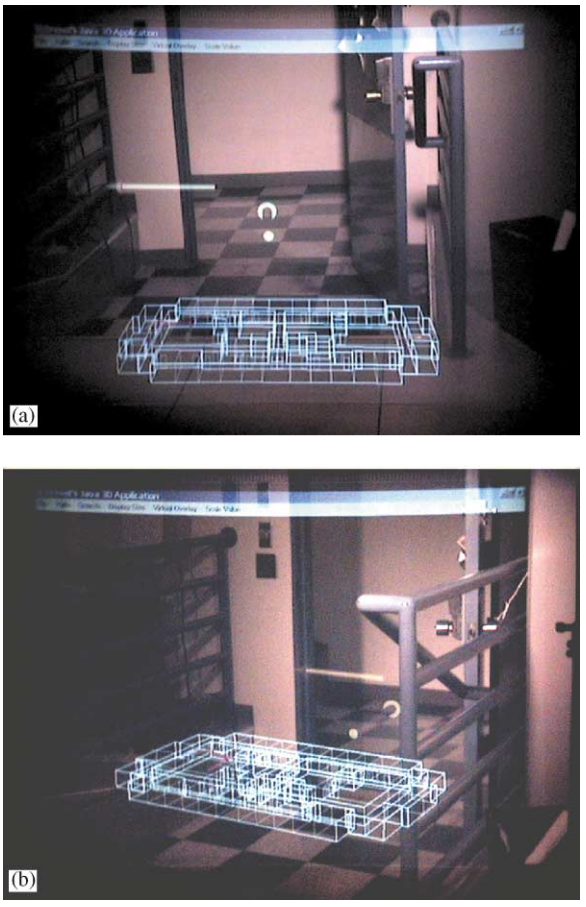

Fig. 6. AR UI in DRM-tracked mode (imaged through see-through head-worn display). (a) A body-stabilized world-aligned WIM with world-space arrows. (b) The same UI with the user at a different position and orientation.

interaction elements) in order to respond to the current tracking accuracy or available display technologies.

Other researchers have also begun to explore how UIs can take into account tracking errors and other environment-specific factors. MacIntyre and Coelho [21] introduce the notion of *level-of-error* filtering for augmented reality: computing a registration error value that is used to select one of a set of alternate representations for a specific augmentation. We believe that their single pose measurement metric needs to be extended to distinguish position errors (which we explore here) from orientation errors, and to account for other varying tracking characteristics (e.g., update rates or likelihood to drift). Butz and colleagues [22] describe an adaptive graphics generation system for navigational guidance. While our projects share many of the same goals, we concentrate on AR UIs, while their initial implementation focuses on small portable devices and stationary displays.

3.1. Complementary tracking modes

Our module assumes different technologies for tracking a user's position in three different circumstances: within part of a research laboratory served by a high-precision ceiling tracker (Fig. 5), in indoor hallways and rooms outside of the ceiling tracker range (Fig. 6), and outdoors (Fig. 7). Orientation tracking is done with an InterSense IS300 Pro hybrid inertial/magnetic tracker. We can track both the user's head and body orientation by connecting head-worn and belt-worn sensors to the unit.

When outdoors with line of sight to at least four GPS or Glonass satellites, our system is position tracked by

an Ashtech GG24 Surveyor dual-constellation real-time-kinematic (RTK) GPS system. For indoor tracking, we use a Point Research PointMan Dead-Reckoning Module (DRM) and an InterSense Mark II SoniDisk wireless ultrasonic beacon. The system can detect whether the beacon is in range of an InterSense Mark II ceiling tracker. The Mark II tracker is connected to a stationary tracking server and the position updates of the roaming user's SoniDisk beacon are relayed to the user's wearable computer using our Java-based distributed augmented reality infrastructure [13].

For indoor areas outside the range of the ceiling tracker, we rely on a dead-reckoning approach that combines a pedometer built into the DRM and an orientation tracker, with environmental knowledge expressed in spatial maps and accessibility graphs [23].

Tracking accuracies and update rates vary widely among these three position tracking approaches. The IS600 Mark II ceiling tracker can track the position of one SoniDisk to a resolution of about 1 cm at 20–50 Hz. Experimental evidence for our dead reckoning approach reveals a typical positional accuracy of 1–3 m. Since the position updates occur in direct response to pedometer activity, the update rate is directly coupled with the user's step frequency (about 1–3 Hz). The outdoor RTK differential GPS system has a maximum tracking resolution of 1–2 cm at an update rate of up to 5 Hz. The GPS accuracy may degrade to 10 cm, or even meter-level when only four or five satellites are visible. If we lose communication to our GPS base station, we fall back to regular GPS accuracy of 10–20 m.

3.2. Adaptive augmented reality user interface

As a test application, we have developed in Java 3D [24] an AR UI for navigational guidance that adapts to the levels of positional tracking accuracy associated with the different tracking modes. Fig. 5 shows a view through the see-through head-worn display when the user is accurately position tracked by the ceiling tracker. The system overlays features of the surrounding room, in this case a wireframe model consisting of our lab's walls and ceiling, doors, static objects of interest (e.g., a rear projection display), and rooms in the immediate neighborhood. Labels are realized as billboarded polygons with transparent textures for the label text (Java 3D Text2D objects). Labels are anchored at their corresponding 3D world positions, so that closer objects appear to have bigger labels. The color scheme highlights important objects (e.g., results of a navigational query and passageways from the current room to the main corridors).

When we roam with our mobile system—away from the ceiling tracker, but not yet outdoors where GPS can take over—we currently depend upon our hybrid, dead-reckoning system for positional data. As a result, we have relatively more accurate orientation tracking than position tracking. To leverage the relatively superior orientation accuracy in this situation, we have chosen to situate much of the overlaid material when roaming within the context of a world-in-miniature (WIM) [20]: a scaled-down 3D model of our environment.

Our WIM has a stable position relative to the user's body, but is oriented relative to the surrounding physical world. That is, it hovers in front of the user, moving with her as she walks and turns about, while at the same time maintaining the same 3D orientation as the surrounding environment of which it is a model. In related work on navigational interfaces, Darken and colleagues [25] explore different ways of presenting 2D and 3D map information to a user navigating in a virtual environment. They conclude that while there is no overall best scheme for map orientation, a self-orienting "forward-up" map is preferable to a static "north-up" map for targeted searches. The WIM is a 3D extension of the "forward up" 2D option in Darken's work. Because our WIM's position is body-stabilized, the user can choose whether or not to look at it—it is not a constant consumer of head-stabilized head-worn display space, and does not require the attention of a tracked hand or arm to position it. If desired, the WIM can exceed the head-worn display's field of view, allowing the user to review it by looking around, since the head and body orientation are independently tracked. The WIM incorporates a model of the environment and an avatar representation of the user's position and orientation in that environment. It also provides the context in which paths are displayed in response to user queries about routes to locations of interest.

When the user moves out of range of the ceiling tracker, position tracking is shifted to the dead-reckoning tracker. To notify the user that this is happening, we first replace the registered world overlay with the WIM model, but at full-scale and properly registered. Then the WIM is interpolated in scale and position to its destination configuration [26]. This animation provides useful information that makes it possible for the user to orient herself with respect to her current position in the WIM. Additional spatial orientation help is provided by the introduction of the avatar, which is highlighted for a few seconds.

Fig. 6 shows the UI just after this transition. Because the head–body alignment is relatively constant between parts (a) and (b), the position of the projected WIM relative to the display is similar in both parts, but the differing position and orientation of the body relative to the world reveal that the WIM is world-aligned in orientation. These images also include route arrows that point the way along a world-scale path to a location that the user has requested (in this case, the nearest stairway). As the user traverses this suggested path, the arrows advance, always showing the two next segments. The

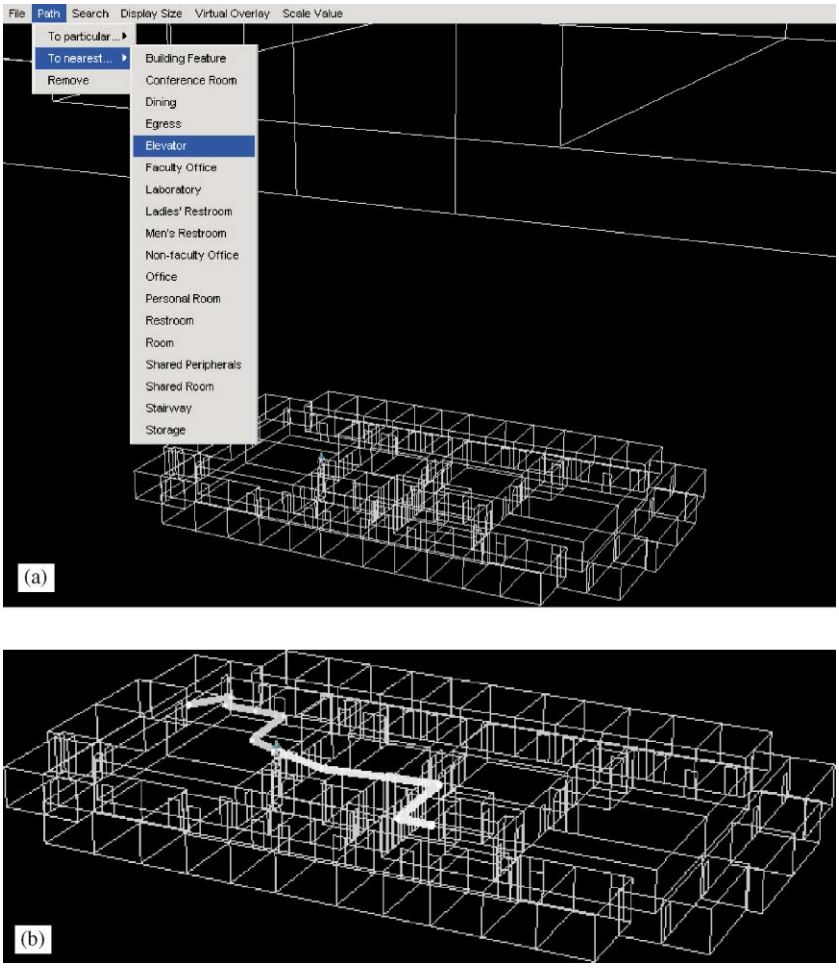

Fig. 8. Navigational guidance. (a) User query. (b) Different solution paths in the WIM.

WIM also displays the entire path, which is difficult to see in these figures because of problems imaging through the see-through head-worn display. (A more legible view of a path is in shown in Fig. 8(b), which is a direct frame-buffer capture, and therefore does not show the real world on which the graphics are overlaid.)

4. View management

No matter how well the information filtering component works, the resulting AR view might still be cluttered and hard to understand. This can occur when augmented material is positioned awkwardly and ambiguously in the user's view. For example, labels and annotations might overlap each other, making them hard to decipher and unclear as to which of several physical objects they annotate. Fig. 7 provides an example of suboptimal annotation placement: The system positions three building labels ("Buell", "St. Paul's Chapel", and "Fayerweather") on top of one real building (*Buell Hall*). This occurs because the simplistic labeling algorithm used for this figure simply places building labels at the screen positions to which the centers of the real-world buildings get projected. Since the centers of these three buildings project quite close to each other and the algorithm does not take into account which parts of buildings are obstructed by other objects, it is not clear which labels refer to what physical building. All that a user who is unfamiliar with the environment can infer, is that all three buildings lie in the particular direction that the labels define.

Label color and other visual attributes can be utilized to denote distance from the user and to emphasize the fact that some buildings are hidden by others [27], but that should only happen when the object to be labeled is completely occluded. Fig. 7 would be much more effective if the label "St. Paul's Chapel" were

Fig. 9. View management (imaged through see-through head-worn display). (a) Head-tracked colleague's head is constrained to be visible to head-tracked observer. (b–c) Therefore, virtual agenda automatically moves to avoid obstructing colleague's head as observer and colleague move.

overlaid on top of the chapel's rotunda, which is clearly visible.

Our *view management* component tries to ensure that annotations correctly refer to the visible parts of the infrastructure as seen from the current viewpoint. Also, it makes sure that annotations do not accidentally occlude each other or other important objects of which the user should be guaranteed a clear view. Fig. 9 illustrates a simple example of a "protected" object that should not be occluded by virtual material. The example application provides support for augmented collaborative meetings [28]. The three images show one meeting participant's view of her colleague, as seen through a see-through head-worn display. Both participants' heads are position- and orientation-tracked and a distributed AR environment provides personalized views of shared 3D graphics models that are discussed during the meeting.

In Fig. 9(a), the observer, whose view is shown, has just brought up a screen-stabilized virtual meeting agenda, which is constrained to be visible to the observer and to be positioned as close as possible to the center of the observer's display. Her colleague's head is constrained to be visible to the observer, as long as it remains within her view frustum. Fig. 9(b) and (c) shows how the agenda automatically moves out of the way to avoid obscuring the colleague's head when either the observer or colleague move. In part (c), it has moved to the other side of the observer's head. For a short transition period during this move, one of the visibility constraints had to be relaxed. In our current framework we experiment with resolving such temporary conflicts by exploiting flexibilities in the way virtual objects are displayed. Possible solutions include moving the flexible object around the protected object swiftly and smoothly while shrinking it in size, or making the object semi-transparent while it smoothly crosses the protected object.

A simple two-element example, such as the one in Fig. 9, is easy to implement, since the system only has to attend to a single protected area. The geometric processing in this case involves only simple comparisons of one upright rectangle representing the agenda's projection on the view plane with upright rectangular extents representing the colleague's head's projection and the viewable area of the head-worn display. Two-dimensional UIs, such as Microsoft Word, already position find/replace dialogue boxes in a similar fashion, such that they do not block the text segments to which they refer. View management becomes significantly more difficult, however, if multiple objects, with different types of constraints, are to be considered. If handled naively, satisfying one constraint by moving an object out of the way of another object is likely to violate other constraints of nearby or associated objects.

To fulfill all requirements posed by the visibility constraints, and to do so in real time, the view management module requires a good representation of the occupied and unoccupied portions of a user's view, which must be updated every rendering frame. We currently make layout decisions for view management in 2D space of the user's projection plane, based on rectangular approximations of the objects' projections [29]. This approach leverages the efficient 2D space-management techniques we developed earlier [30], making it possible for our view management algorithm to perform at interactive speed.

Fig. 10 shows a scene that our view management module manages in real time: The application is a meeting situation like the one described above. Here, the participants are meeting to discuss the design of our campus model. Building labels are laid out dynamically for each participant so that each label overlaps only its own building as seen from that person's view. Labels change size and style depending upon the amount of space available. In this case, the user selected the model of *Buell Hall* to inspect, causing a copy of the building to be made, and information about it to appear in an attached document that is constrained to stay close to the building copy. Like the agenda in the top left corner,

Fig. 10. View management in a collaborative system (imaged through see-through head-worn display). Labels are laid out dynamically to annotate the buildings of a campus model as seen by the observer. UI elements avoid overlapping the colleague's head and the campus model.

the building copy and document avoid overlapping other objects determined to be more important (e.g. the campus buildings and the colleague's head).

5. Conclusions and future work

We have presented three UI management techniques that we believe are crucial for creating effective MARS UIs. *Information filtering* selects only the information most relevant to the user, her current task, and her current context. *UI component design* chooses a suitable format in which this information is to be presented to the user, based on available resources and tracking accuracy. *View management* attempts to ensure that virtual objects are laid out appropriately in the field of view such that they do not occlude more important physical or virtual objects and that their relationships with other objects are unambiguous. These mechanisms have been implemented and tested as separate modules in our MARS development environment. We are in the process of integrating all three into a single system.

There are many directions in which we would like to extend the work reported on here.

Determining meaningful subjective properties in our filtering framework (importance and task vectors) is of critical importance to the utility of the goal- and context-based filtering modes. These properties are highly domain-specific and, generally, hard to derive. We would like to develop specific guidelines for domain analysis to make it easier for our domain experts to fit the domain expertise into a format that is compliant with the filtering model. User studies need to be conducted to verify the usefulness of the model for the different domains for which it is used.

We are working on a more flexible and automated solution for UI component design. To accomplish this, we will need to derive a taxonomy of MARS components that can be combined to form different AR interfaces that provide the user with the same functionality under radically different context conditions. Our next step beyond adapting to differences in tracking accuracy will be accommodating different display technologies.

In our work on view management, we are interested in exploring how a rule-based system could control the view-management component in response to changes in the users' environments and tasks. This would eliminate the need for the users of this component to impose visibility constraints explicitly on the components of the MARS UI. User studies will have to be carried out to determine the kind(s) of automatic layout control that will work best for users engaged in different tasks.

Finally, the domain-specific user context models underlying the information filtering, UI component design, and view management steps should be unified, with the resulting common representation forming the basis for decisions in all of the steps needed to create effective, informative, and enjoyable MARS UIs.

Acknowledgements

The research by Columbia University described here is funded in part by Office of Naval Research Contracts N00014-99-1-0249, N00014-99-1-0394, and N00014-99-0683, NSF Grant IIS-00-82961, and gifts from Intel, Microsoft, and Mitsubishi. The research by the Naval Research Lab described here is funded in part by the Office of Naval Research.

References

[1] Furness T. The super cockpit and its human factors challenges. Proceedings of the Human Factors Society 30th Annual Meeting. Santa Monica, CA, 1986. p. 48–52.
[2] Fuchs H, Livingston M, Raskar R, Colucci D, Keller K, State A, Crawford J, Rademacher P, Drake S, Meyer A. Augmented reality visualization for laparoscopic surgery. Proceedings of the First International Conference on Medical Image Computing and Computer-Assisted Intervention. October 1998.
[3] Feiner S, Webster A, Krueger T, MacIntyre B, Keller E. Architectural anatomy. Presence 1995;4(3):318–25.
[4] Feiner S, MacIntyre B, Seligmann D. Knowledge-based augmented reality. Communications of the ACM 1993; 36(7):52–62.

[5] Hoff WA, Lyon T, Nguyen K. Computer vision-based registration techniques for augmented reality. Proceedings of the Intelligent Robots and Control Systems XV, Intelligent Control Systems and Advanced Manufacturing, Vol. 2904. November 1996. p. 538–48.

[6] Caudell T, Mizell D. Augmented reality: an application of heads-up display technology to manual manufacturing processes. Proceedings of the Hawaii International Conference on Sys Sci Hawaii, January 1992.

[7] Webster A. Feiner S, MacIntyre B, Massie W, Krueger T. Augmented reality in architectural construction, inspection and renovation. Proceedings of the ASCE Third Congress on Computing in Civil Engineering. Anaheim, CA, 1996. p. 913–19.

[8] Feiner S, MacIntyre B, Höllerer T, Webster T. A touring machine: prototyping 3D mobile augmented reality systems for exploring the urban environment. Proceedings of the International Symposium on Wearable Computers. Boston MA, October 1997.

[9] Höllerer T, Feiner S, Pavlik J. Situated documentaries: embedding multimedia presentations in the real world. Proceedings of the ISWC '99 (Third International Symposium on Wearable Computers). San Francisco, CA, October 18–19, 1999. p. 79–86.

[10] Thomas B, Close B, Donoghue J, Squires J, De Bondi P, Morris M, Piekarski W. ARQuake: an outdoor/indoor augmented reality first person application. Proceedings of the ISWC '00 (Fourth International Symposium on Wearable Computers). Atlanta, GA, October 16–17, 2000. p. 139–46.

[11] Julier S, Baillot Y, Lanzagorta M, Brown D, Rosenblum L. BARS: battlefield augmented reality system. NATO Symposium on Information Processing Techniques for Military Systems. Istanbul, Turkey, October 2000.

[12] Baillot Y, Gagas E, Höllerer T, Julier S, Feiner S. Wearable 3D graphics for augmented reality: a case study of two experimental backpack computers. NRL Technical Report, 2000.

[13] Höllerer T, Feiner S, Terauchi T, Rashid G, Hallaway D. Exploring MARS: developing indoor and outdoor user interfaces to a mobile augmented reality system. Computers and Graphics 1999;23(6):779–85.

[14] Shneiderman B. Designing the user interface., 3rd ed. Reading, MA: Addison-Wesley, 1998.

[15] Cheverst K, Davies N, Mitchell K, Blair GS. Developing a context-aware electronic tourist guide: some issues and experiences. Proceedings of the CHI '00. Netherlands, 2000.

[16] Hull R, Neaves P, Bedford-Roberts J. Towards situated computing. Proceedings of the ISWC '97 (First International Symposium on Wearable Computers). Cambridge, MA, October 13–14, 1997. p. 146–53.

[17] Benford S, Fahlén L. A spatial model of interaction in large virtual environments. Proceedings of the ECSCW '93, Milan, Italy, September 1993.

[18] Benford S, Bowers J, Fahlén L, Greenhalgh C, Mariani J, Rodden T. Networked virtual reality and cooperative work. Presence 1995;4(4):364–86.

[19] Julier S, Lanzagorta M, Baillot Y, Rosenblum L, Feiner S, Höllerer T, Sestito S. Information filtering for mobile augmented reality. Proceedings of the ISAR '00 (International Symposium on Augmented Reality). Munich, Germany, October 5–6, 2000. p. 3–11.

[20] Stoakley R, Conway M, Pausch R. Virtual reality on a WIM: interactive worlds in miniature. Proceedings of the Human Factors in Computing Systems (CHI '95). May 7–11, 1995. p. 265–72.

[21] MacIntyre B, Machado Coelho E. Adapting to dynamic registration errors using level of error (LOE) filtering. Proceedings of the ISAR '00 (International Symposium on Augmented Reality). Munich, Germany, October 5–6, 2000. p. 85–8.

[22] Butz A. Baus J, Krüger A, Lohse M. A hybrid indoor navigation system. Proceedings of the IUI 2001 (International Conference on Intelligent User Interfaces), Santa Fe, NM, January 14–17, 2001. p. 25–32.

[23] Höllerer T, Hallaway D, Tinna N, Feiner S. Steps toward accommodating variable position tracking accuracy in a mobile augmented reality system. Second International Workshop on Artificial Intelligence in Mobile Systems (AIMS '01) 2001.

[24] Sowizral H, Rushforth K, Deering M. The Java 3D API specification. Reading, MA: Addison-Wesley, 1997.

[25] Darken R, Cevik H. Map usage in virtual environments: orientation issues. Proceedings of the IEEE VR '99. 1999. p. 133–40.

[26] Pausch R, Burnette T, Brockway D, Weiblen M. Navigation and locomotion in virtual worlds via flight into hand-held miniatures. Proceedings of the SIGGRAPH '95. 1995. p. 399–401.

[27] Kamada T, Kawai S. An enhanced treatment of hidden lines. ACM Transactions on Graphics 1987;6(4):308–23.

[28] Butz A, Höllerer T, Feiner S, MacIntyre B, Beshers C. Enveloping users and computers in a collaborative 3D augmented reality. Proceedings of the IWAR '99 (International Workshop on Augmented Reality). San Francisco, CA, October 20–21, 1999. p. 35–44.

[29] Bell B, Feiner S, Höllerer T. View management for virtual and augmented reality. Proceedings of the ACM UIST 2001 (Symposium on User Interface Software and Technology). Orlando, FL, November 11–14, 2001.

[30] Bell B, Feiner S. Dynamic space management for user interfaces. Proceedings of the ACM UIST 2000 (Symposium on User Interface Software and Technology). San Diego, CA, November 5–8, 2000. p. 239–48.