An Augmented Virtual Reality Interface for Assistive Monitoring of Smart Spaces

Shichao Ou, Deepak R. Karuppiah, Andrew H. Fagg, Edward Riseman
Department of Computer Science,
University of Massachusetts,
Amherst, MA - 01003, USA
chao,deepak,fagg,riseman@cs.umass.edu

January 6, 2004

Abstract

In large scale surveillance applications, coherent presentation of data coming from myriad sensors becomes a problem. For example, tasks such as "locate an intruder" are no longer easy when the user is facing a room of monitors connected to hundreds of cameras. Therefore, there is a need for a system that allows the user to easily navigate the data space. Due to the scale of the application, such systems should also be robust with respect to hardware and software failures, as well as to varying bandwidth conditions.

Our strategy is to first build software units that provide sensor abstractions (e.g., location of a person, noise level of a predefined region) to lessen the burden of dealing with individual sensors from the user. Second, we project this abstract sensor information into an augmented virtual reality interface for presentation to the user. The AVR interface offers a common medium to display abstract information constructed from multiple sensors, as well as allowing access to raw sensor information such as video streams, or a mixture of both. Also, the AVR interface can synthesize views not serviced by the physical cameras. The sensor abstraction and the smooth transition between sensors enable the user to intuitively navigate the data space. Further more, through this interface, the user can dynamically reconfigure the system resources. We will demonstrate three scenarios highlighting the above mentioned features.

1. Introduction

With the availability of cost effective sensors and processors, pervasive sensor systems with hundreds of sensors are now becoming a reality. As a result, applications now have to deal with the explosion of sensor information. In the context of surveillance, the traditional *user interface* (UI) , such as a room of monitors each showing a live video stream from a corresponding camera, does not scale as the number of sensors grows.

In traditional UIs, switching between different camera streams on a single monitor is unavoidable when there are fewer monitors than cameras. Switching across many cameras can become extremely confusing when following an event of interest. For example, figure 1 shows a cluttered scene from the perspective of 3 different cameras. Many people find locating the person circled in Camera 2's view (the middle picture) in the other two cameras quickly difficult.[1] This image example only has 3 cameras, we can imagine this problem becomes even more evident if we are monitoring 20 monitors connected hundreds of cameras. The delay in the user's reaction is caused by the fact that the cameras are located in different corners of the room, or in other words they are *spatially disconnected*. When abruptly switched from one camera viewpoint another, one needs time to regain the comprehension of elements of the surrounding environment (this is called *situation awareness* [3]). The "spatial disconnect of sensors" problem undermines the user's ability to achieve *situation awareness* quickly. The importance of *situation awareness* is even more apparent when the sensors are steerable, e.g., cameras mounted on mobile robots or pan-tilt units. Only when the user acquires a good spatial sense of the current viewpoint can he efficiently operate such sensors remotely. More importantly, without a continuous transition between sensors, the user would need to remember a sequence of events using symbolic names such as "in room 250" or "I saw this event through camera 315.". These symbolic names do not explicitly convey any spatial relationships. For longer term memorization, the user would have to mentally establish spatial relationship between these symbolic names. This requires more effort from the user, drives up training costs and in-

[1] In the first camera view, the target is in the upper right corner of the image (partially occluded, far away from the camera); in the third camera view, he is in the middle of image.

Figure 1: The "spatial disconnect of sensors" problem with the traditional "room of monitors" user interface - try to quickly locate the person circled in the middle picture) in the other two camera views (different perspective of the same scene). The answer can be found in the footnote.

creases response time to time-critical events.

Our approach to these problems is to first build software units that provide sensor abstractions. For instance, by grouping sets of cameras to track a person or multiple persons in a space and reporting only the 3D location of the tracked object, we summarize several camera image streams into one abstract sensor - the tracked object location. The user can now simply focus on the tracked feature, rather than trying to analyze the scenes from individual cameras simultaneously. Like real sensors, we would like our abstract sensors to work reliably. However, large scale sensor networks are prone to routine failures influenced by hardware, software, or the environment. Therefore, in order for sensor abstraction units to work reliably, we populate the environment with redundant sensors. By having multiple overlapping sets of sensor abstraction units, we can easily switch units or reconfigure the current unit to recover from failures.

With sensor abstraction available, a suitable interface is needed to convey the abstract sensory information to the user and to assist the user in directing the data gathering process. A number of research efforts in the aviation and military domains have shown that better understanding of terrains can be achieved by navigating through 3D interfaces [11, 9, 1]. Cockburn and Mckenzie [2] have shown spatial arrangement of documents allows for rapid document retrieval. The gaming industry long ago moved from 2D to 3D to provide a richer and more immersive world that the players can freely explore. These provide evidence to the assertion that since we live in a 3D world, the most intuitive way to interact with remote spaces is through a 3D virtual environment. Through a virtual world, the user is able to explore the spatial configuration of the environment, engage his natural abilities to interact with the environment and construct internal cognitive maps of the space [6]. Therefore we implemented an Augmented Virtual Reality (AVR) interface. A virtual scene is used to display abstract information in 3D, and is augmented with real-life video streams when necessary. Such an interface offers a common medium to display abstract information (e.g. system status) in a spatially relevant manner.

In the upcoming sections, we will discuss in detail the framework we use for constructing abstract sensors to summarize information in a scene in a fault tolerant manor. We will show how an AVR interface can be an efficient medium that enables the user to navigate the sea of data in a seamless fashion, as well as to reconfigure system resources to recover from faults. Examples will be given to demonstrate the features of our system.

2. Fault Containment Unit Hierarchy

In small scale applications, the user has no trouble monitoring individual sensors. However, this is not true in large scale applications when the user has to deal with thousands of sensors (e.g., cameras, motion sensors, heat senors or mobile robots). In order to alleviate the burden on the user from having to specifically deal with individual sensors, some form of sensor abstraction is needed. We can achieve this by building high-level software modules that can address top-level task objectives. For instance, tasks such as "track any person when they enter this region," or "alert me when there is any noise above such threshold in this region" can be specified and allow the system to deal with how to best allocate sensors and resources to accomplish the task. This delegation requires the system to be robust with regard to various low-level hardware and software failures. In our implementation, we have adopted an existing robust sensor abstraction and fault-tolerant framework called a "Fault Containment Unit" (FCU) [5].

Complex systems are designed using redundant resources with the expectation that failures caused by some

subset of resources can be overcome by other subsets. We formalize this dynamic reallocation of redundant resources in the event of failures by defining a Fault Containment Unit. A *Fault Containment Unit* is a construct of software modules with a task specification, a set of resources needed to accomplish its task and a set of built-in rules for handling failures. In the extreme case where fault containment is not possible, the containment unit communicates a status report to its instantiating process. Also, a fault containment unit with a higher level task specification can have subordinate containment units as its resources. Thus, by forming a hierarchy of containment units we can perform various tasks in our smart space such as localization of people and robots, and recognition of people. In our smart space, faults occur due to failure of hardware (sensors, robots, CPU), software (algorithms, controllers), and communication.

Below we give a glimpse of how fault containment units are used to manage resources in our system. Although in this example we specifically talk about camera controllers, the framework can be easily extended to other types of sensors and actuators such as acoustic sensors or mobile robots. Two low level controllers that run on our pan-tilt-zoom (PTZ) cameras are the saccade controller (ϕ_s) that moves the camera towards the direction of an interesting feature e.g. motion, and the foveate controller (ϕ_f) that brings the feature of interest to the center of the field of view. Figure 2 shows a schema that can perform the saccade followed by the foveate task. This schema also generates reports that describe its own behaviour like *hardware fault, no target, target lost and heading report*. If a target feature is detected and foveated, then the sensor achieves state $X1$ where the feature is actively tracked. As long as the actions of the foveation controller preserve this state, a heading to the feature is reported. In all other cases, an appropriate report describing the nature of failure is generated.

When two instances of the SACCADE-FOVEATE FCUs are simultaneously in state $X1$ and they are driven by features derived from the same subject, then there is often sufficient information for triangulating the location of the subject. A higher level containment unit called LOCALIZE FCU receives the event streams generated by two subordinate SACCADE-FOVEATE FCUs under its management and produces a report regarding the location of the subject. The subject of interest may at times be moving or stationary. In the former case, the LOCALIZE FCU may have to actively manage the subordinate FCUs, while in the latter case it can instantiate a MONITOR FCU that just confirms the presence of the stationary feature in the last known location.

At the highest level a FCU supervisor may instantiate multiple LOCALIZE FCUs each of which are responsible for maintaining a robust track of a single subject of interest. Over time, the event streams coming from lower levels

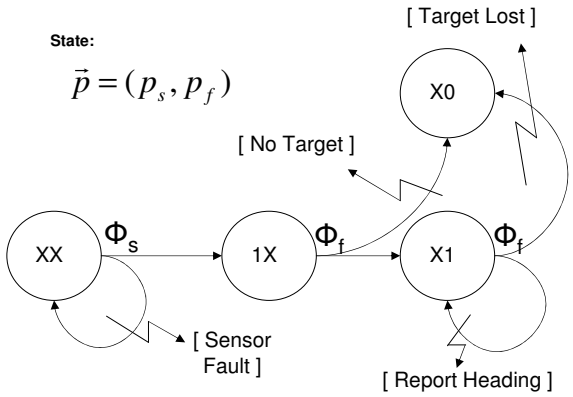

Figure 2: Fault Containment Unit wrapper for the saccade-foveate model. This schema performs the saccade followed the foveate task. ϕ_s is a saccade controller, while ϕ_f is a foveate controller. XX, $1X$, $X1$, $X0$ are possible states of the controllers. The schema generates reports that describe its behaviour such as no target, and heading report. Errors are handled through the hierarchy of FCUs.

are used to build and update a collection of features that describe each subject. When a LOCALIZE FCU reports a lost track, the annotation of features to the corresponding subject is handed off to a new instantiation of the LOCALIZE FCU with a different set of resources that are best placed to take over the tracking. This hierarchy is shown in figure 3.

2.1. User as top level in Fault Containment Unit hierarchy

When the number of resources available to a containment unit is large, there is an exponential increase in the choices for resource allocation. Therefore, offline hand-coding or prioritizing different courses of actions is quite difficult. Furthermore, these methods will fail when an unforeseen error occurs and the predefined FCUs cannot recover from the fault. Therefore we propose to add user to the highest level of the hierarchy since humans can react to situations using prior experience and reconfigure a FCU to recover the system from a fault state. This removes the need to presuppose all but the most routinely anticipated contexts. Figure 3 shows the user acting as the top level in the FCU hierarchy. The user interface provides a way to convey information abstracted by the FCU hierarchy as well as a direct means to interact with it.

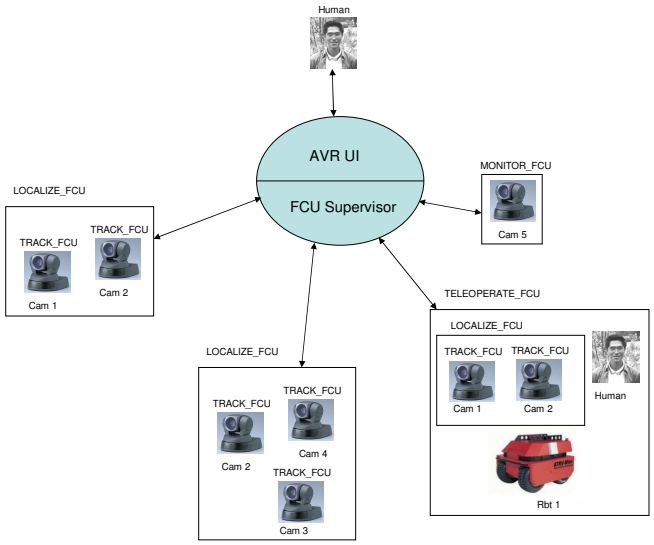

Figure 3: Fault Containment Unit Hierarchy. At the center of the hierarchy is the FCU supervisor. In this example, it is responsible for instantiating multiple LOCALIZE FCUs to maintain a robust tracking of a subject of interest. When a LOCALIZE FCU reports a lost track, a new LOCALIZE FCU with a different set of resources is instantiated to take over tracking. When the subject is stationary, a MONITOR FCU is instantiated to simply confirm the presence of the stationary feature in the last known location. When FCU supervisor fails to recover from a fault state, through an user interface, the user is able to dynamically reconfigure FCUs to help the system recover from fault. For instance, he can recruit a mobile robot to form a new TELEOPERATE FCU.

3. Augmented Virtual Reality Interface

With FCUs working to collect abstract sensory information, a suitable interface is needed to both convey this information effectively to the user and to enable the user to direct the information gathering process. In the context of large scale surveillance, the spatial disconnect of sensors makes it difficult for the user to quickly acquire an understanding of the surrounding environment after a sensor switch. Therefore, delaying the reaction time when the user is following a stream of events. The new interface must also deal with this issue. Spatial cognition studies have shown navigating through 3D interfaces can help the user to better understand the environment [11, 9, 1], as well as to access information faster [2]. Therefore, we implemented an Augmented Virtual Reality (AVR) interface, in which a virtual scene displays abstract information in 3D, and the virtual scene is augmented with real-life video streams when necessary. The proposed AVR interface works in 3 modes:

- a pure virtual world mode that displays abstract information extracted by the sensors,
- a full video stream mode that streams real videos from each camera, and
- a mixed mode that overlays partial video streams from the real world on top of the virtual world. The virtual world conveys the context and the spatial relationships within the scene, while the dynamic window displays the real-time imagery of scene (see figure 4).

3.1. The Virtual Reality Mode

Augmented Virtual Reality (AVR) is not a new concept. According to the taxonomy of mixed reality by Tamura and Yamamoto [10], AVR belongs to the definition of Class B (Video see-through) or Class C (On-line tele-presence) depending on whether the real world imagery comes from scenes directly in front of the user or a remote site. In fact, our implementation falls into the category of class C - Online tele-presence.[2] Tamura and Yamamoto [10] also described an application for the mixed-reality interface - a virtual shopping mall where the shopping mall is a virtual environment while the sales items are displayed in high definition images. This application creates a virtual shopping experience that is close to a real-life experience where the shopper can freely walk between stores for items they desire.

Similar to the virtual mall experience, a virtual environment interface in the context of surveillance is immersive, i.e., the user can freely move about without abrupt spatial changes. As a result, the user can hop between islands of "sensory information hubs" while maintaining the sense of spatial relationships between sensors. In our AVR interface implementation, the user can monitor the scene from the viewpoint of a certain camera using the Full-screen Video streaming as in the traditional UI's case. However, when a camera switching event occurs (either initiated by the user or by the system), the scene seamlessly fades into a virtual world scene that is in-sync with the real-life scene. Then the user "flies" from the view of the source camera (in the

[2] merging video images transmitted from a remote site and virtual images, giving the observer a mixed view of two different different worlds

virtual scene) to that of the next camera (this is called a *virtual fly-through* (Figure 6b)). Lastly, the scene transitions back to the real-life video streaming. The smooth transitions using *virtual fly-through* enables the user to synthesize those views that are not serviced by real world cameras. Therefore, the user is able to move between the sensors while maintaining his spatial relationship with the objects and events happening in the world. This is very important for achieving and maintaining *situation awareness* and allows the user to exploit his natural spatial cognition abilities.

A continuous virtual work space provides a common medium for displaying real world information (video streams), abstract sensor information as well as system state information (e.g. "lost track" or "intruder found"). With a virtual environment interface, information can now be stored and accessed spatially. For example, missed events can be stored at the occurrence location and can then be accessed later for analysis. This naturally engages the user's spatial memory, rather than forcing the user to remember a room number or camera ID, and therefore can reduce the cognitive load on the user.

Traditional monitoring systems generally have dedicated bandwidth requirements in order to stream full screen videos. However, in certain situations dedicated bandwidth may not always be available. For example, in a scenario where security personnel equipped with wireless mobile computer are pursuing a potential intruder, access to full video streams may not be possible. With a virtual environment interface, network bandwidth requirements can be greatly reduced. Only abstract information such as (x,y,z) coordinates, or the color of the tracked subject, are sent across the network. These are lighter weight representations than the raw video data stream. Moreover, the virtual environment offers the flexibility of using different levels of detail when presenting information to the user, thus avoiding information overload. For example, when multiple subjects moving about an area are being precisely tracked, the system does not need to display these avatars in the interface unless any of the subjects move into a restricted area. However, when tracking-quality deteriorates, the user can switch back to full video streaming mode to ensure tracking.

3.2. Full screen Video Mode and Mixed Mode

When monitoring from a fixed camera view, depending on the context, the user may choose either the full screen video mode or the mixed mode to continue monitoring the scene. The full screen video mode works the same way as the traditional UI. However, in our system full screen video mode can also be used to display abstract information extracted by the sensors by overlaying bounding boxes around the tracked object to attract attention of the user.

A major advantage of mixed mode is that it enables the user to monitor the environment in both the abstract information space and the real space. Such a mix takes advantage of the best of both interfaces. For example, this is useful: 1)when the user wishes to keep an eye on a person and a mobile robot at the same time. Since he cares about the motion of person, a video stream is necessary. However, for the robot he just needs to be aware of its location. In an AVR interface, the person can be monitored through the overlay video clips while the robot is displayed as a virtual avatar in the VR scene. This way, when the robot moves behind a desk, in a VR scene, the desk can either become transparent or simply disappear to allow the user to maintain track of both objects without needing to switch to another camera view. 2) Using a cleaned-up VR scene as backdrop, one can overlay more abstract information on the screen without overloading it. For example, text overlays can be easily perceived on a clean VR scene than on a cluttered video scene.

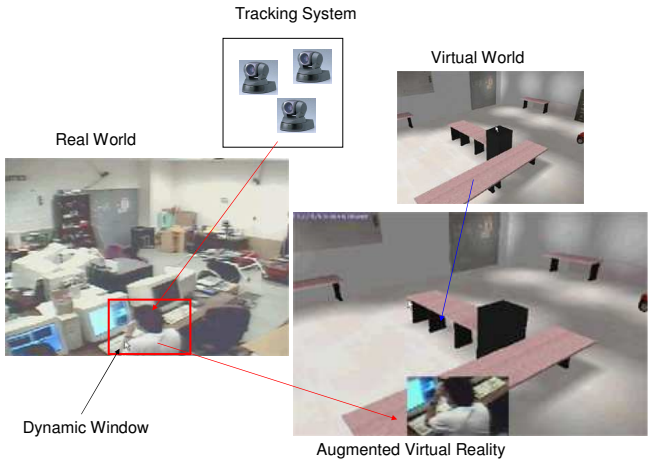

Figure 4: Mixed mode - Using information extracted by the tracking system, the AVR interface overlays the real world imagery of the tracked object on top of the virtual world.

4. System Implementation

The UMass Smart Space has five Sony PTZ EVI-D100 cameras mounted on the walls and an ATRVJr mobile robot equipped with a fixed camera. Our computer rack consists of a cluster of six VMIC single board computers each with a 928 Mhz processor and 256 MB RAM. The nodes in the cluster share a 100Mbps and a 1000Mbps ethernet link and a wireless access point to communicate with the robot. Using the NDDS real-time publish-subscribe middleware [8], each node acts as a server of video and track information extracted from the camera. To create the virtual version of the

smart space, room dimension and camera position measurements were taken by hand. Prominent objects such as tables were placed in virtual space roughly in alignment with the placement of the real objects. The robot control interface was implemented using Player/Stage [7]. The AVR interface was implemented using Genesis3D [4].

At startup, the AVR interface renders the virtual world, and subscribes to sensors for relevant information from the smart space (e.g., roll-pitch-yaw angles and zoom level from the cameras). The virtual camera models in the AVR interface are thus synchronized with their real world counterparts. This feature enables smooth transitions between real and virtual views. When the user requests a full video stream from any camera in the AVR interface, a subscription is activated to the corresponding video stream. Moving object locations published by the FCU supervisors are rendered in the AVR user interface as avatars or as partial video streams in the mixed mode. Other abstract information such as system state (e.g., "lost track" events) and how to recover the field of view of the tracked object can be overlayed on top of the AVR interface. The user also can remotely operate robots in the smart space using this interface. Each robot's state is updated using both its published odometry as well as its track information from the FCU supervisor.

5. Example Tasks

We present three real-life scenarios in which our system was tested to highlight the efficacy of our system in these situations.

The first scenario demonstrates the summarization and abstraction of imagery from 4 live video streams into the 3D location of a tracked object using an appropriate design of FCUs. We also show how the coupling of the FCU sensor abstraction with the AVR interface can help the user easily keep track of an event of interest. Our smart room is equipped with 5 PTZ cameras. There is no single pair of cameras that can cover the entire room. In order to robustly track a moving object, two overlapping containment units FCU1 and FCU2 were instantiated with two cameras each (Figure 5). In VR mode, rather than manually going from camera to camera to make sure there is nothing moving about in the space, the user can now simply request that any moving subjects be tracked and monitor the abstracted object location that is rendered on the AVR interface. Furthermore, when one FCU fails to track the target, an error signal is raised through the FCU hierarchy and the hierarchy automatically switches to the other FCU to maintain tracking. These events are transparent to the user. On the interface side, when the tracked object moves out of the field of view of the current camera, the assistive AVR interface automatically suggests a next camera (with a white arrow) for the user to switch to in order to maintain field of view of the target (Figure 6).

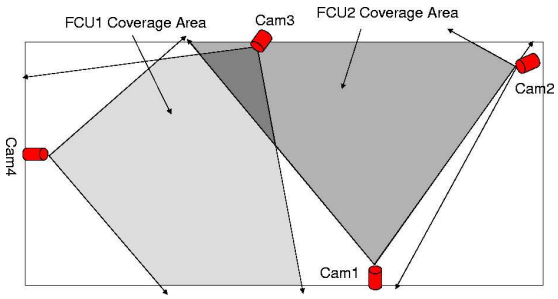

Figure 5: This illustrates the design of two FCUs to cover most of the reachable space within a rectangular room. This enables the system to reliably tracking moving subjects in this room. Two containment units FCU1 and FCU2 are instantiated with two cameras each. The light-shaded and darker-shaded overlays indicate the valid triangulation regions for FCU1 and FCU2 respectively. The two FCU regions overlap each other (highlighted with the darkest shade). In general, the cameras do not have fixed orientation, and other FCU instantiations are possible.

The second scenario shows that using the AVR interface the user can operate in both the real world and virtual world: monitor a person's motions through the partial real life video stream window while remote driving a robot in virtual mode. Thus, even when the robot becomes occluded in the real video stream, the user can still find and operate it due to the transparent display of the AVR interface (Figure 7).

The last scenario shown in Figure 8 demonstrates the dynamic reconfiguration of a FCU with user intervention in the FCU hierarchy. (a) Initially, the smart space tracks a moving person, who later tries to avoid the tracking system by hiding under a table, out of the view of all the cameras. When the system loses track of the person, it notifies the user since it is unable to recover from this fault by itself. (b) Playing the assistive role, the user reacts by teleoperating the cameras, switching between cameras and tries to recover the lost object, but fails since the intruder is hiding beyond the view of any wall-mounted cameras. (c) After arriving at the target camera, the user switches to the full video mode and attempts to locate the missing person. (d) Not finding the lost person, the user decides to recruit a mobile robot into the current the FCU in attempt to recover from fault. He teleoperates the mobile robot to explore the vicinity of the last tracked location using views from the robot-carried camera and different wall-mounted cameras. The important thing to note here is that due to the smooth transitions, the user is able to maintain a good spatial sense of the environment, and therefore is able to operate the robot without any delay after each camera change. Finally, he uncovers the person hiding under the desk and thus recovers the system

from the fault state. The system resumes tracking of the subject. We chose to teleoperate the robot to demonstrate the flexibility and the smooth transitions of the interface. It is also possible to simply recruit the robot and have it autonomously scan the area where the system lost track of the subject. This scenario demonstrates the achievement of successful tracking fault containment through the use of AVR interface, and the effectiveness of placing the user in the loop.

The videos for the above scenarios can be accessed at `http://www-robotics.cs.umass.edu/Research/Distributed_Control/PerComm04/`

6. Conclusions and Future Work

In the context of large scale surveillance applications, this paper introduces a novel Augmented Virtual Reality user interface to convey information to the user effectively. The AVR interface is an ideal medium to display abstract information as well as raw sensor information, or a mix of both. Also, it has the advantage of being able to synthesize views not serviced by the physical cameras using virtual reality. The smooth transition between sensors enables the user to intuitively navigate the data space. The Fault Containment Units framework is used to build robust, fault tolerant abstract sensors that also alleviates the user's burden of dealing with myriad individual sensors. This allows the user to focus more on solving high-level problems, especially unforeseen system errors, in which humans are more capable than machines. This also in turn helps to strengthen the robustness of the entire system since we can treat the user as the top-most level of the Fault Containment Hierarchy. This system helps users to be more efficient at handling time-critical events such as locating and containing an intruder by reconfiguring system resources online. We have demonstrated the use of our system with three real-life scenarios.

User interaction provides valuable dynamic control information for efficient reactions to urgent unanticipated situations. For future work, we would like such information be learned by the system, allowing interaction in similar contexts to be minimized. We plan user studies to evaluate the effectiveness of the AVR interface under real-life large scale sensor network conditions. We would also like to explore the usefulness our system in other pervasive computing application such as remote touring.

Acknowledgments

The authors gratefully thank professor Rod Grupen for his support and comments on this paper, and acknowledge support for this work from DARPA/ITO DABT63-99-1-0004(MARS), NSF CDA 9703217(Infrastructure) and NASA NAG9-1445#1.

References

[1] W. Barfield. and C. Rosenberg. Judgements of azimuth and elevation as a function of monoscopic and binocular depth cues using a perspective display. *Human Factors*, 37(1):173–181, 1995.

[2] A. Cockburn and B. McKenzie. Evaluating the effectiveness of spatial memory in 2d and 3d physical and virtual environments. In *Proceedings of CHI*, 2002.

[3] M. R. Endsley. Towards a theory of situation awareness in dynamic systems. *Human Factors*, 37(1):32–64, 1995.

[4] Eclipse Entertainment. Genesis3d. `http://www.genesis3d.com`.

[5] D. R. Karuppiah et al. Software mode changes for continuous motion tracking. In P. Robertson, H. Shorbe, and R. Laddaga, editors, *Self-Adaptive Software*, volume 1936 of *Lecture Notes in Computer Science*, Oxford, UK, April 17-19 2000. Springer Verlag.

[6] A. H. Fagg, S. Ou, T. R. Hedges, M. Brewer, M. Piantedosi, P. Amstutz, A. Hanson, Z. Zhu, and R. Grupen. Human-robot interaction through a distributed virtual environment. Workshop on Intelligent Human Augmentation and Virtual Environments, 2002.

[7] B. P. Gerkey and R. T. Vaughan. Most valuable player: A robot device server for distributed control. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001)*, pages 1226–1231, 2001. `http://playerstage.sourceforge.net`.

[8] Real-Time Innovations Inc. Ndds: The real-time publish-subscribe network. `http://www.rti.com/products/ndds/literature.html`.

[9] M. John, H. Oonk, and M. Cowen. Using two- dimensional and perspective views of terrain. Technical Report 1815, SPAWAR Systems Center, 2000.

[10] H. Tamura and H. Yamamoto. Vision and graphics in producing mixed reality worlds. In *IEEE and ATR Workshop on Computer Vision for Virtual Reality Based Human Communications*, pages 78 –85, Jan 1998.

[11] C. Wickens, C. Liang, T. Prevett, and O. Olmos. Egocentric and exocentric displays for terminal area navigation. In *Human Factors and Ergonomics Society 38th Annual Meeting*, pages 16–20, 1995.

(a)

(b)

(c)

(d)

Figure 6: Example task 1 - robust tracking with FCUs. In this scenario, two overlapping containment units FCU1 and FCU2 were instantiated with two cameras each (Figure 5). (a) The user gives the command "track any moving target in this room". The system automatically chooses the appropriate FCU in order to maintain an accurate tracking of the subject; (b) the abstracted object location, as well as other abstract information (e.g. FCU ID) are rendered on the AVR interface. When the tracked object moves out of the field of the view of the current camera, the assistive AVR interface automatically suggests a next camera (the white arrow at the bottom) for the user to switch in order to maintain field of view of the target; (c)*virtual flythrough* during the camera switch allows the user to keep track of the subject and maintains his spatial sense during entire transition process; (d) after the camera switch, the user is able to monitor the rest of the motion of the subject.

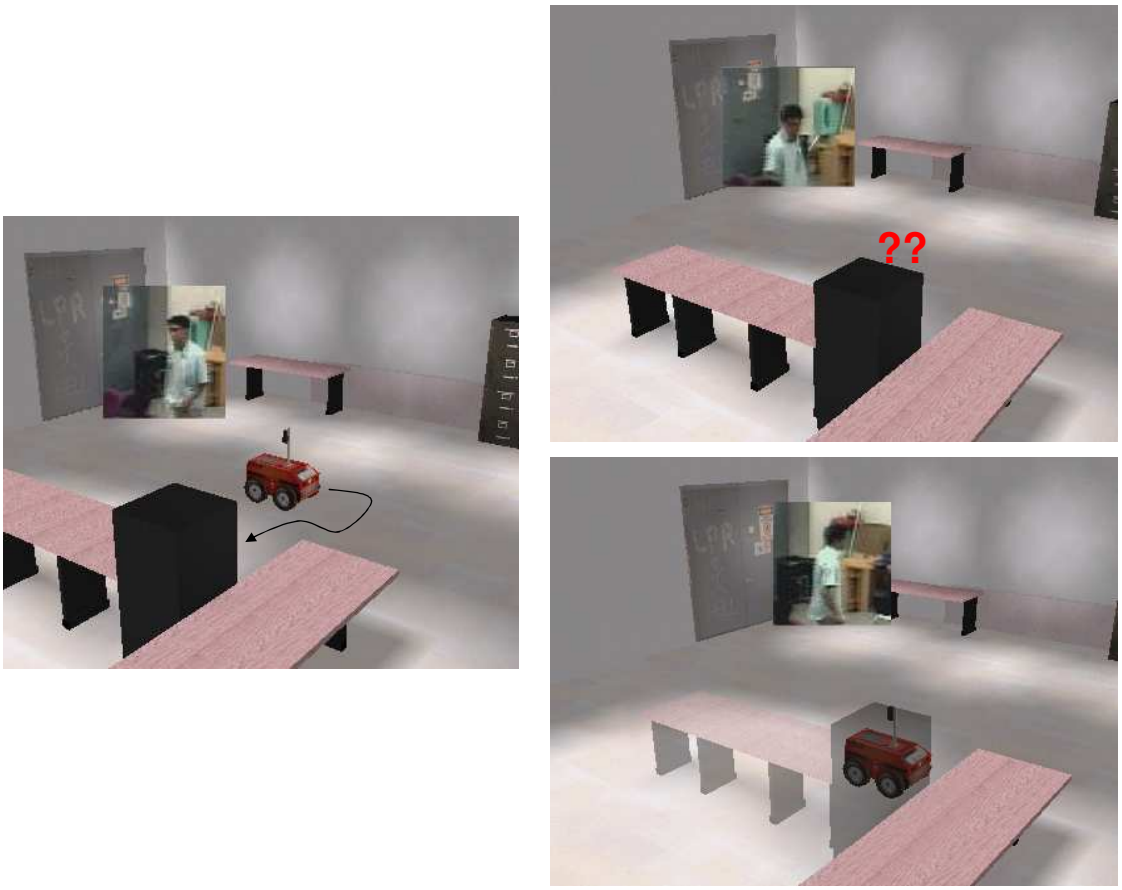

Figure 7: Example task 2 - advantage of mixed mode. Using the AVR interface the user can work in both the real and virtual worlds: in this scenario, the user is remotely driving a mobile robot, while monitoring a person's motion through the mixed mode's partial video stream window. Later, the user loses sight of the robot due to occlusion. Without needing to switch to another camera, the user is able to find the robot thanks to the transparent display of the AVR interface.

(a) (b)

(c) (d)

Figure 8: Example task 3 - dynamic reconfiguration of FCU with user intervention in the FCU hierarchy.
(a) The smart room tracks a moving person, who tries to avoid the tracking system by ducking down under a table, out of the view of all the cameras. When the system loses track of the person, it notifies the user since it is unable to recover from this fault by itself; (b) Playing the assistive role, the user reacts by teleoperating the cameras, switching between different streaming modes and tries to recover the lost object, but is unable to do so. (c) This figure shows the transition from the pure virtual mode to full video mode and attempt to locate the missing person; and (d) Not finding the lost person, the user decides to recruit a mobile robot into the current the FCU in attempt to recover from fault. He teleoperates the mobile robot to explore the vicinity of the last tracked location using views from the robot-carried camera and different wall-mounted cameras. The important thing to note here is that due to the smooth transitions, the user is able to maintain a good spatial sense, and therefore is able to operate the robot without any delay after each camera change.