Shared Interactive Video for Teleconferencing

Chunyuan Liao[1], Qiong Liu[2], Don Kimber[2], Patrick Chiu[2], Jonathan Foote[2], Lynn Wilcox[2]
[1]Dept. of CS, Univ. of Maryland
[2]FX Palo Alto Laboratory, 3400 Hillview Ave. Bldg. 4, Palo Alto, CA 94304, U.S.A.
Phone: 1-650-813-6957
{liu, kimber, chiu, foote, wilcox}@fxpal.com

ABSTRACT
We present a system that allows remote and local participants to control devices in a meeting environment using mouse or pen based gestures "through" video windows. Unlike state-of-the-art device control interfaces that require interaction with text commands, buttons, or other artificial symbols, our approach allows users to interact with devices through live video of the environment. This naturally extends our video supported pan/tilt/zoom (PTZ) camera control system, by allowing gestures in video windows to control not only PTZ cameras, but also other devices visible in the video image. For example, an authorized meeting participant can show a presentation on a screen by dragging the file on a personal laptop and dropping it on the video image of the presentation screen. This paper presents the system architecture, implementation tradeoffs, and various meeting control scenarios.

Keywords
Collaborative device control, video enabled device control, gesture based device control, panoramic video, video communication, video conferencing, distance learning.

1. INTRODUCTION
Remotely participating in meetings is increasingly popular for purposes such as distance learning and project discussions between distant areas and countries. Through remote participation, students can attend classes from their dorms, scientists can participate in seminars held in other countries, and executives can discuss critical issues without leaving their offices. This paper focuses on interactive video techniques to support meetings and teleconferences. For the purposes of discussion, we will define "local" as the physical meeting room, and "remote" as anywhere else.

MM'03, November 2-8, 2003, Berkeley, California, USA.

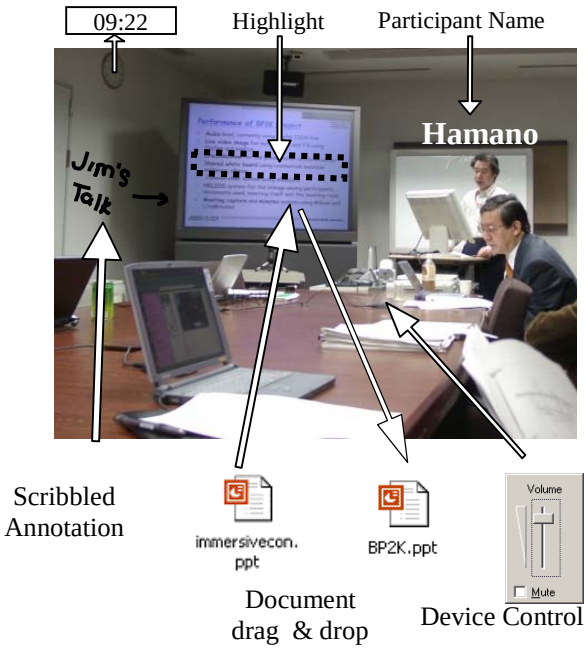

Figure 1. Scenarios for using shared interactive video in a teleconference.

In the literature, "interactive video" appears with several different meanings. Zollman considers it to be any video where the user has more than minimal on-off control over what appears on the screen, including random access, still frame, step frame, and slow play [16]. Tantaoui et al. use the term similarly in their paper [14]. VideoClix™ sees interactive video as hotspot enhanced recorded video [15]. Darrell's "interactive video environment" refers to a virtual reality system where people can interact with virtual creatures [3]. Aside from these, a relatively common usage in the distance-learning field is two-way video/audio/data communication shared among participants [9]. A more formally specific definition of interactive video, from [12] is: "A video application is interactive if the user affects the flow of the video and that influence, in turn, affects the user's future choices." The meaning we adopt in this paper is very close to the union of the last two definitions. More specifically, we view interactive video as a live video

interface that allows us to affect and control real-world devices seen in the video.

We described a live video supported camera control system in an earlier paper [11]. There, users are presented with a panoramic video overview, and may select regions of that window for closer inspection in a close-up video window. In other words, the panorama video overview serves as the control interface for mouse actions that steer a Pan/Tilt/Zoom (PTZ) camera (which provides the high-resolution close-up view).

A natural extension is to let users control other devices using related mouse or pen based gestures performed in video windows. For example, an authorized participant may drag a presentation file from a desktop to a remote presentation screen seen in the video window, causing that presentation to be remotely displayed. To facilitate this,, the video may be augmented with metadata indicating the controllable devices, or annotations such as the names of participants.

Our work differs from the device control system proposed in [8], in that it does not require specific devices for control, nor does it require people to be in the controlled environment. Unlike the "graspable interface" that allows people to manipulate virtual objects with real "bricks" [4], our system operates real world equipment through the live video. Moreover, in contrast to the VideoClix™ or iVast™ products [15][7] that support metadata editing of recorded video, our system works on live video and controls real-world devices.

Live video as a reference cue for power-plant control has been explored in [13]. Goldberg et al. use video as reference cue to convey "Tele-Actor," a skilled human who collects the video and perform actions during navigation [6]. To our knowledge, more general video-enabled device control for meetings has not been presented in the recent literature. By using a video interface to control devices in a meeting environment, meeting participants may easily overcome the limitations of their physical locations. For example, a remote meeting participant may use a mouse to draw figures on a local presentation screen from 2000 miles away. As we illustrate in Figure 1, a remote participant may also drag a copy of a given presentation from the video image of the (local) screen to a remote desktop, check the local time, or choose among microphones for better sound quality. If the remote person wishes to give a presentation, they may drag a presentation file onto the video image of the (local) presentation screen, and adjust the screen brightness for proper visual effect.

Our current implementation of this idea supports "interactive video" by augmenting the live video captured by our FlySPEC subsystem with device-control messages. Currently, it supports useful file operations like printing and local/remote file transfer.

The rest of the paper is organized as follows. In the next three sections, we present the system overview, the video acquisition system, and the video canvas and metadata. In section 5, we present system deployment and several application scenarios. Summary and future work are detailed in Section 6.

2. SYSTEM OVERVIEW

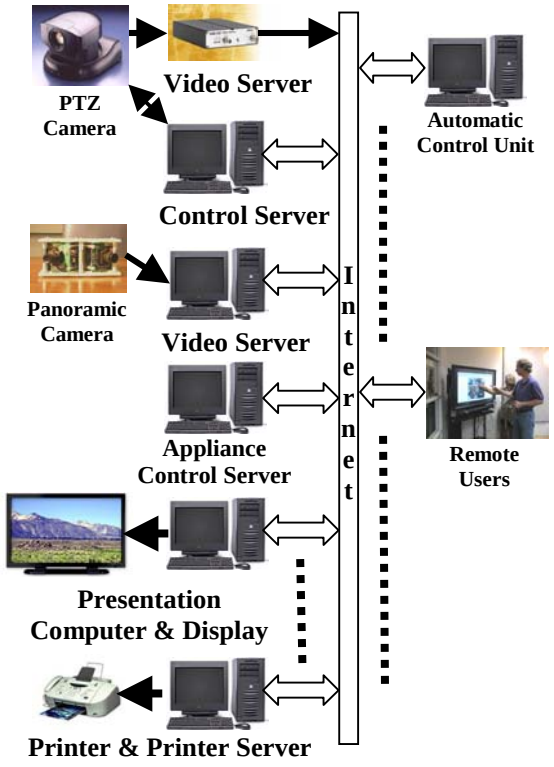

Figure 2. Connecting multiple control components through the Internet.

A basic idea in our system is the "Video Canvas," which associates and orients all images, annotations and metadata. A Video Canvas can be thought of as a shared blackboard onto which the cameras paste images and from which users request views, control devices seen in the views, and draw annotations to be shared with other users. Each interface provides an overview of the canvas, and a close-up view of some region. A meeting participant may control their close-up view by circling a region of interest in the overview. Furthermore, they may add annotations such as "name-tags" and digital ink marks, or transfer files using drag and drop operations.

Figure 2 shows the system architecture. Video from the controlled scene is captured using a hybrid camera that includes both a PTZ and a panoramic camera. Videos from both cameras are sent to users as sequences of JPEG

images in real time. A camera control server running on a networked workstation controls the PTZ camera through a RS-232 serial link. The camera control server accepts HTTP control requests through the Internet/Intranet, and sends pan/tilt/zoom commands to the PTZ camera. Given the camera constraints, the camera control server optimally satisfies users with a combination of optically and digitally zoomed video.

An appliance-control server manages appliances other than the cameras. This server translates control requests into specific device control commands. When this control server receives requests from remote users, it sends the appropriate commands and data to the local devices. With the architecture of Figure 2, we can add arbitrarily many cameras, screens, printers, and other devices in the system.

As seen on the right side of Figure 2, remote clients connect to the system through networks. Remote users can control the FlySPEC camera and other devices from these remote clients. They can also immediately see the effects of control actions through the video. We have implemented several different client applications specialized for different uses, described in Section 5.

3. THE VIDEO ACQUISITION SYSTEM

The video acquisition system can be described in four parts: the camera hardware, the graphical interface for user interaction, the camera management system, and the network video service.

3.1 The FlySPEC Camera

Figure 3. The FlySPEC Camera

The video for our device control environment is provided by a FlySPEC system [11]. Figure 3 shows a FlySPEC camera. This is a hybrid camera constructed by combining a PTZ camera and a panoramic camera called FlyCam [5]. The panoramic camera covers a wide-angle view, and provides low-resolution video for the entire video canvas. The PTZ camera can provide high-resolution video of smaller areas or capture details of small objects. Requested video views are provided by the PTZ camera, or by electronic pan/tilt/zoom from the panoramic camera. The close proximity of the panoramic and PTZ cameras makes it easy to find the correspondence between the PTZ view and a region in the panoramic view with a simple 2D affine transformation. This correspondence is useful for controlling the PTZ camera based on low-resolution panoramic video. It also allows seamless switching of the video source between the panoramic and PTZ cameras.

3.2 Graphical user interface for controlling cameras and other devices

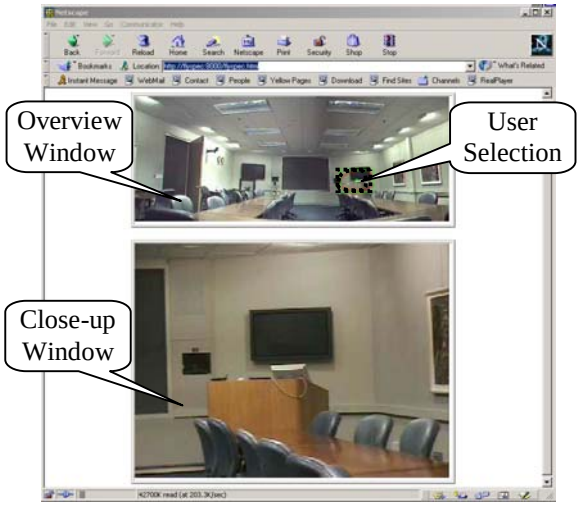

Figure 4. Web-based Graphical User Interface for PTZ Camera Control and General Meeting Device Control

To facilitate both device and camera control, we designed the graphical user interface shown in Figure 4, which is based on the interface described in [11]. In the web browser, the upper window shows a resolution-reduced video from the panoramic camera, and the lower window shows the close-up video produced by the FlySPEC system. Using this interface, the user adjusts the close-up video by circling an interesting region in the panoramic view with the mouse. The region inside the circled area's bounding box will then be shown in the close-up view window. So instead of confusing buttons for pan and zoom direction, an intuitive gesture selects the desired camera view. In addition, we define a richer set of gestures for interacting with other varied devices, as discussed later in Section 5.

3.3 The Camera Management Approach

Details of our most recent camera management algorithm are described in [10]. Relative to the camera image plane (corresponding to the video canvas) is an 'ideal video signal' $f(x,y,t)$ where (x,y), are canvas coordinates and t is

time. $f(x,y,t)$ is the idealized video signal available from a perfect unlimited resolution camera. Because ideal cameras are not generally available, $f(x,y,t)$ is represented by a band-limited signal estimated from the panoramic and PTZ camera. The camera management goal is to steer the PTZ camera so that $\hat{f}(x,y,t)$ is a good approximation to $f(x,y,t)$ in the most highly viewed regions. Let $p(x,y,t|I)$ be the probability that users will want to see details around (x,y), conditioned on control inputs I (which can include user view requests as well as features computed automatically by the system). The main idea of the camera management approach is to minimize the expected distortion during a period T, given by

$$D[\hat{f}, f] = \iiint p(x,y,t \mid I)\left|\hat{f}(x,y,t) - f(x,y,t)\right|^2 dxdydt \qquad (1)$$

by moving the PTZ camera to an optimal position. After the PTZ camera takes the best possible image, each meeting participant who requests an interesting view will be served independently according to his/her selection and the available details of the scene.

3.4 The Video Service

The NTSC video from the PTZ camera is connected to an off-the-shelf motion JPEG server made by Axis®. The server compresses NTSC video into a sequence of JPEG images in real time, and serves them over the Internet at a maximum rate of 30 frames per second. Multiple client requests slow the frame rate gracefully. To serve a large number of video users, a high performance server can be installed between the motion JPEG server and the Internet. Since the prototype system does not serve a large number of users at this early stage, connecting the motion JPEG server directly to the Internet provides reasonable system performance.

The panoramic camera of the FlySPEC camera is connected to a workstation that can capture video from fixed cameras and stitch these video inputs into a panoramic video in real-time. A computer server running on this workstation can also send panoramic videos or close-up videos in JPEG sequences to remote users according to their requests.

Server push of the motion JPEG requires higher bandwidth than some solutions such as MPEG, but has the advantage of simplicity, and low latency. In our system, low latency is critical important for device-control performance. The pushed JPEG images may be easily displayed in browsers capable of displaying server pushed images, or using simple Java applets.

4. VIDEO CANVAS AND METADATA

In our system, the video canvas is a core concept for integrating various software components. It can be thought of as a shared blackboard onto which images are pasted and user views are extracted, and onto which annotations and metadata may be anchored. For convenience, the canvas-coordinate system is defined according to the panoramic video image coordinates, and used for all metadata and view control information.

4.1 Definition of hot spots and metadata

To operate devices (e.g. screen, speaker, or printer) appearing in live video, we need to model these devices with either a 3-D model or a 2-D model [13]. In the case of static objects and fixed FlySPEC cameras, it is easy to locate the objects the user is operating by simple "within-region" checking in the canvas. In the case of a dynamic environment, where objects' images move freely on the canvas, the interaction becomes complicated since we need to recognize and trace these object in real-time, which is still an open problem. In our system, we use a simple 2-D model and assume that the FlySPEC camera is fixed and that objects remain static or change infrequently enough that manual updating of positions is feasible.

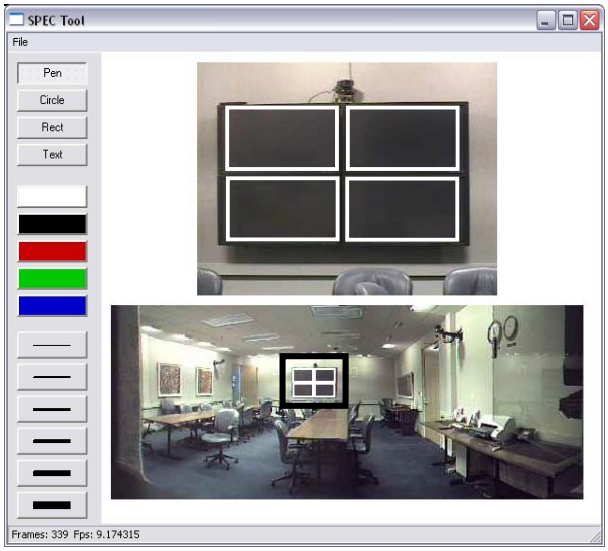

Figure 5. Canvas annotation tool, used to define hotspots and metadata.

The annotation tool shown in Figure 5 can be used to create and edit metadata such as annotations or hotspots for supporting various interactions with visible devices. It provides video windows showing both panoramic and close-up views. Either window can be used for gestures to control the camera, produce annotations, or define metadata such as hotspots associated with various controllable appliances. For simple annotations, basic drawing primitives for scribbles, text, rectangles, etc. are provided. Other metadata can be added using popup menus to indicate the type of an object, and drawing primitives to define its region. This is typically done in the close-up view for greater precision.

Metadata is organized in a class hierarchy of object types.

Entity
- Scribble
- Person
  - Speaker
- Appliance
  - Display
  - Printer
  - …

Objects have attributes and methods that are inheritable through the class hierarchy. The attributes include the extent, which is the region covered by the object, defined by a rectangle or poly-line, and auxiliary information such as URLs of informative web pages. Methods define behaviors such as responses to user interface actions like mouse events. The following is pseudo code for a typical object:

```
Controllable Entity {
  X, Y;              // coordination in the virtual video canvas
  Region;            //extent of this object, defined
                     // by a rectangle or poly-line
 Type;               // Object type: txt annotation, scribble
                     // annotation, hotspot
  ContainsPoint(x,y);        // Used for hit detection
  Draw(context, mode);
  OnMouseIn(event);          //handlers for mouse events
  OnMouseOut(event);         // …
 ….
}
```

Figure 5 illustrates the definition of metadata objects corresponding to controllable displays. Each of the white rectangular regions defines one of the displays. Note that these rectangles appear in both the panoramic and close-up windows, which may both be considered as views onto the same video canvas. In the NoteLook application we will describe later, users can drag slides from the hotspots corresponding to these displays and make notes on them. For a unified way to handle hotspot interaction, a special hotspot for users' local desktop is introduced to support the file drag and drop operation between the remote devices and the local folder.

4.2 Device Control via Hotspots

When multiple people discuss a project in a meeting room, people frequently share drawings on a common area, such as a white board or a large poster. If drawings are accepted by a display hotspot, the display server will take drawings from users and display the result. This functionality can facilitate shared-drawings for both remote participants and onsite participants. Participants with a PC or PDA can draw figures directly in the video window without walking to a shared-screen. The system can attach labels or other annotations to the live video. For example, when we hold a videoconference between a US group and a Japanese group, it sometimes difficult for us to remember every colleague's name in another group. We tackle this problem by manually attaching names to sitting participants. Figure 6 shows an example of this application.

Figure 6. Canvas annotation tool, used to define hotspots and metadata.

4.3 The Video Canvas and Video Views.

To support direct operations on a close-up view, we need to form the mapping between the panoramic view and the close-up view. Since FlySPEC can provide the current corresponding rectangle of the close-up view in the panoramic image, we can use an approximate linear mapping to get the hot spot rectangle in the current close-up view.

Consider a window of width W and height H showing a close-up view of the region centered at x,y and of width w and height h. Then a point in that view with view coordinates (v_x, v_y) may be mapped to a corresponding point in the canvas with canvas coordinates (c_x , c_y) by:

$$c_x = x + v_x * (w/W)$$
$$c_y = y + v_y * (h/H)$$

and points may be mapped from canvas coordinates to view coordinates by the inverse mapping. In this way, mouse events or annotation data drawn in one view may be

mapped to the canvas coordinate system, and then to any other view. This is demonstrated in Figure 5, which shows rectangular metadata defining four different displays, seen in both the panoramic overview, and the close-up view. Mouse events can also be mapped this way to determine when mouse events occur within various hotspots.

5. SYSTEM DEPLOYMENT AND APPLICATION SCENARIOS

As our prototype, the meeting device control system is deployed in a mid-sized corporate conference room. To provide sufficient video views of the conference environment, multiple FlySPECs were installed at different locations in the conference room. Figure 7 is a top view illustration of the meeting room layout. In this meeting environment, FlySPEC1 is normally used to cover on-site meeting participants. FlySPEC2 is used to cover the conversation between the presenter and on-site participants. FlySPEC3 is mainly used to cover the presenter and presentation screens.

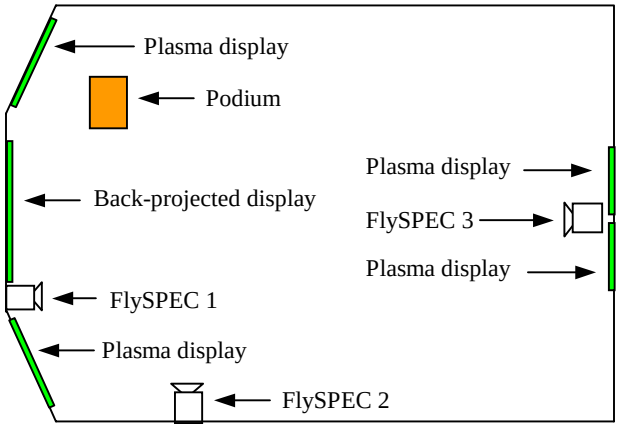

Figure 7. Top view of the meeting room layout

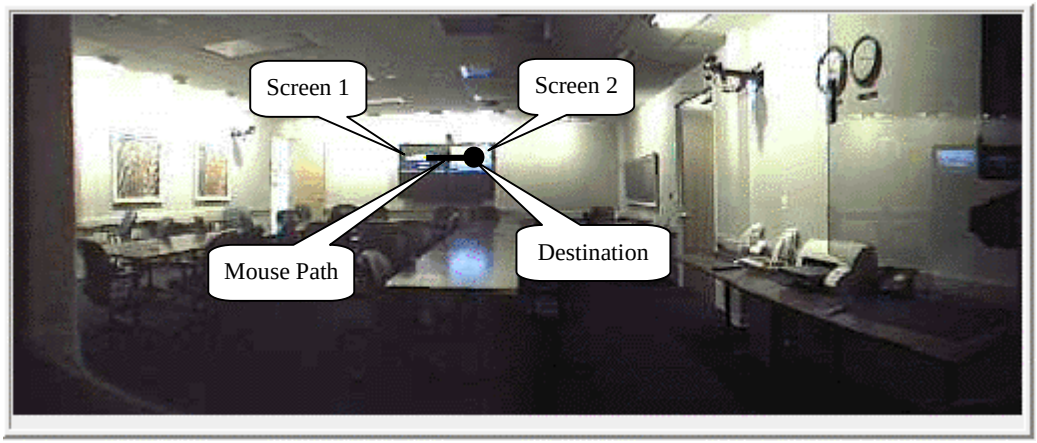

Figure 8. Dragging a slide from screen 1 to screen 2.

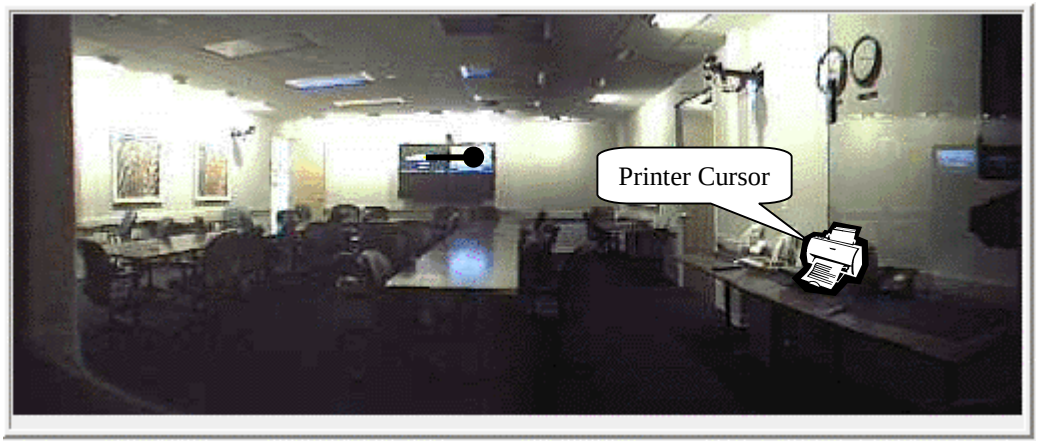

Figure 9. When the mouse cursor moves to the hot spot defined for the printer, it is changed to printer shape.

5.1 Web Browser Supported Interface

To encourage people to use our system in their daily lives, we first implemented a web-based client interface as that shown in Figure 4. In this interface, the overview video window was implemented as a Java Applet. This Java Applet can analyze users' gestures and send proper web

requests corresponding to these gestures. In our implementation, we define the left-mouse-button activated gesture as camera control gesture, and the right-mouse-button activated gesture as general device control gesture. In other words, when a user draws a path with his/her left-mouse-button down, the path will be mapped to a camera control command. When a user draws a path with the right-mouse-button down, the path will be mapped to a general device control command.

With this web-based implementation, a user can drag presentation slides among screens shown in the same video window. More specifically, if we want to drag the slide from screen1 to screen2, we may press the right-mouse-button on screen1, and release the mouse button on screen2. Figure 8 shows the overview window when we try to move a slide from screen 1 to screen 2. The black line and the black dot are marks the applet shows in the window. The callouts are only used to assist our explanation, and are not parts of our interface.

To achieve this drag-and-drop functionality, our system has to run a lightweight server on each presentation computer for slide management. When a presentation is performed in the meeting room, the presenter's slides are first translated into a sequence of JPEG images in a file directory, and the names of these JPEG images are used by the whole system to reference individual slide. After the slide translation, names of all slides are loaded to a screen server to form a stack. Slides are managed as stacks by all screen servers, and only the top slide in a stack can be displayed on a screen. With this mechanism, the presenter and some authorized participants are free to move slides across screens for project discussions. When a user drags a slide from a source to a destination, the gesture is first interpreted by the Java Applet, and the applet will send a http request to the appliance control server. After the appliance control server receives the http request, it will send control commands to corresponding devices. When a destination screen server receives a display command and the name of a specific slide, the name of the slide will be put on the top of the server stack. On the other hand, the source screen server will remove the top of its stack and select the next available slide as the top slide.

Screens can be both source and destination during the drag-and-drop operation. By slightly modifying the screen server, the system can support meeting participants to drag files to the hot spot of an onsite printer. For example, the printer in our deployment can be found on the right parts of Figure 8 and Figure 9. With this functionality, users of our system can conveniently print out slides in the meeting room for records.

5.2 Handling Desktop Files

Even though the web-based interface ensures the basic accessibility to our system, it is restricted by Java applet limitations from handling some operations such as dragging a file from a desktop to a presentation screen. We tackled this problem by developing a wrapper application using COM Object techniques. The application loads the webpage (described in a previous section) in its window. A meeting participant can still use the gestures defined for the web-based interface. To activate extended gestures, such as dragging a file from the desktop to a presentation screen, the applicant should double click the mouse in the application window but outside of the video window. The cost of using these extended gestures is a lightweight application on a user's computer.

Figure 10. When the mouse icon moves to the hot spot for the screen, the icon is changed to hand shape. This hand-shape icon indicates that the user is allowed to manipulate the screen.

This application supports dragging files to remote screens, loudspeakers, or printers. It also supports dragging a file from the screen to an audience member's desktop when the presenter of the seminar has an interesting file shown on the screen. Via the overview window or the close-up window, the user can immediately see his/her control results.

In this implementation, we also added some intuitive cursors to remind users the available functions. More specifically, when a user moves the mouse cursor in the

overview video window or the close-up video window, the shape of the cursor will be changed according to the hot-spot region it covers. In Figure 9, when the mouse cursor moves to the hot spot for a printer in the overview window, it is changed to printer shape. In Figure 10, when the mouse cursor passes the hot spot for the screen in the close-up window, it is changed to a hand-shape to remind the user of screen operations.

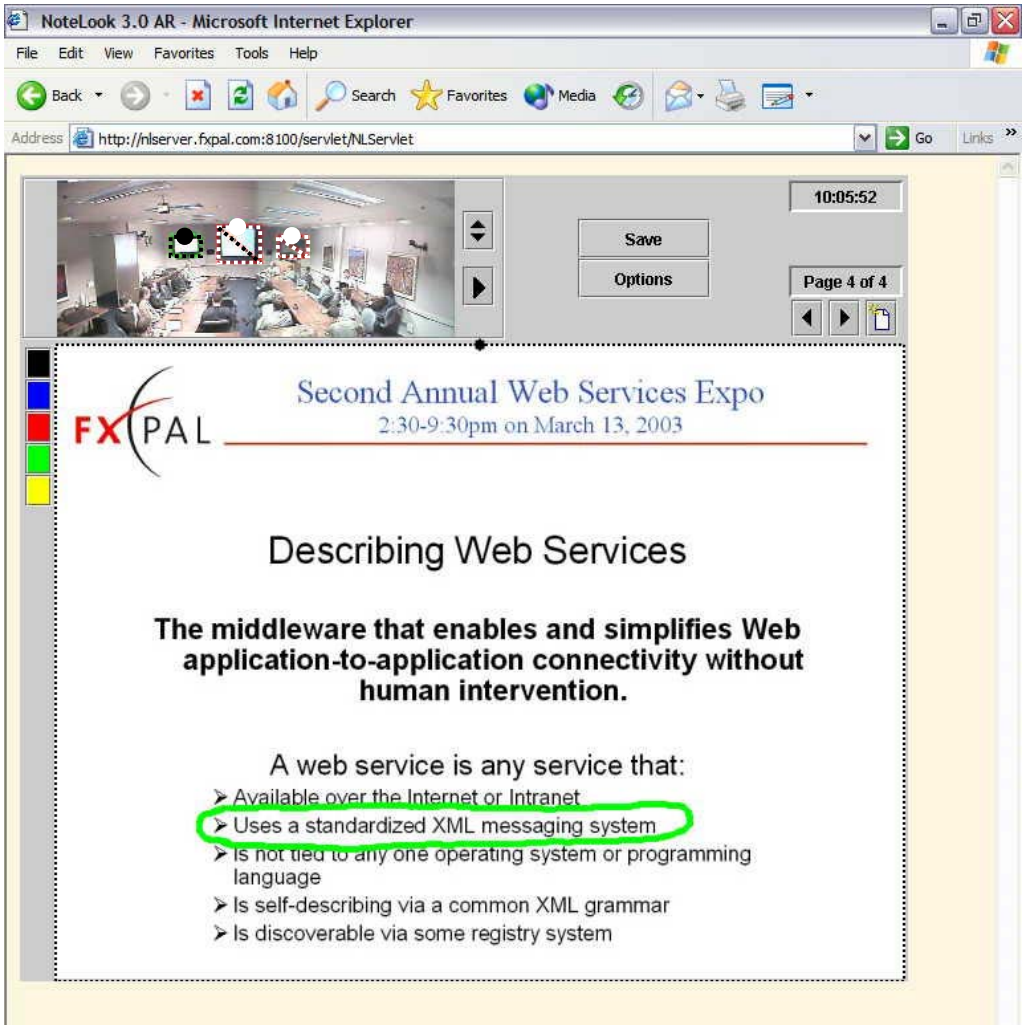

Figure 11. NoteLook 3.0 Screen Shot.

5.3 NoteLook Application

We also incorporated some basic screen control techniques into our NoteLook 3.0 pen-based application to support annotating slides in our multi-display meeting environment [1, 2]. NoteLook facilitates meeting participants in taking notes with personal devices, such as - TabletPC, laptop, or a remote desktop PC. It allows meeting participants to grab a slide or take an image of the meeting and make annotations on it. In this system, all slides and camera shots are handled as JPEG, GIF, or PNG images. Figure 11 shows our most recent NoteLook 3.0 interface. In this interface, a live panoramic video of the meeting room is shown on top of the screen. On the right side of the video window, a spin button is used to cycle through multiple panoramic video sources. The 4 dashed boxes (three appear in the video window and one surrounds the note-taking page) are hot spots that can support gesture-based screen operations. The big dot on top of each dashed box is the handle for moving slides from that hot spot, while the slash in a hot spot bans a user from dragging slides from other hot spots to it. With this interface, a user can move a slide from the first hot spot to the second by dragging the handle of the first hot spot and put it in the second hot spot that authorizes this operation. For example, a user can drag the slide shown on the center public screen to the note-taking area. After making annotations on the slide (as shown in the figure), the user can drag this annotated slide to the left side public screen for discussion.

6. SUMMARY AND FUTURE WORK

We have presented the design and implementation of a meeting-room device-control system based on interactive video-. With this system, an offsite or onsite meeting participant or presenter can easily manage presentation screens, notes, and onsite printing by performing gestures over live meeting videos. By extending this idea, we can define more hot spots associated with other appliances in the scene, and give remote users more control of the conference room. For example, we can define audio-monitor-hot-spots, and drag a music file to an audio-monitor-hot-spot to play music in the conference room. Or, we can also define light-hot-spots and turn on/off lights by clicking them.

Currently, we hardwired all hot spots in our system. In the future, we want to set up a hot-spot definition mode and bind basic control functions with predefined icons. When a system administrator sets the system to the hot-spot definition mode, video windows and all control icons will be shown to the administrator side-by-side. Then the administrator can select a region with a mouse and drag all corresponding control functions to that region with the mouse.

We also want to attach transmitter to various objects, and use sensors to detect locations of those objects. With this approach, hot spots can be defined dynamically. When the hot spots can be defined dynamically, remote users are allowed to control moving devices.

Defining hot spot automatically is another interesting research branch. If we can use image patterns of some basic devices to find these devices in the video automatically, hot spots may be defined automatically. Even though state-of-the-art vision techniques still cannot do a perfect job on object recognition, if an administrator can overwrite the definition, this technique is still feasible and can save people's time.

Finally, it will be ideal if our device-control system can be installed on a mobile platform, and be used in various changing environments.

7. ACKNOWLEDGEMENTS

The authors would like to thank Jonathan Helfman for his helpful comments to this paper.

8. REFERENCES

[1] Chiu, P., Kapuskar, A., Reitmeier, S., and Wilcox, L. NoteLook: Taking notes in meetings with digital video and ink, *Proceedings of ACM Multimedia'99*. ACM Press, pp. 149-158.

[2] Chiu, P., Liu, Q., Boreczky, J., Foote, J., Fuse, T., Kimber, D., Lertsihichai, S. and Liao, C. Manipulating and annotating slides in a multi-display environment, *Proceedings of INTERACT '03*, to appear.

[3] Darrell, T., Maes, P., Blumberg, B., Pentland, A.P. A Novel Environment for Situated Vision and Behavior, *MIT Media Lab Perceptual Computing Technical report*, No. 261, (1994).

[4] Fjeld, M., Ironmonger, N., Voorhorst, F., Bichsel, M., & Rauterberg, M. Camera control in a planar, graspable interface, *Proceedings of the 17th IASTED International Conference on Applied Informatics (AI'99)*, pp.242-245.

[5] Foote, J., and Kimber, D. FlyCam: Practical Panoramic Video, *Proceedings of IEEE International Conference on Multimedia and Expo*, vol. III, pp. 1419-1422, 2000.

[6] Goldberg, K., Song, D.Z., and Levandowski, A. Collaborative Teleoperation Using Networked Spatial Dynamic Voting, *Proceedings of IEEE, Special issue on Networked Robots*, 91(3), pp. 430-439, March 2003.

[7] iVast™, iVast Studio SDK™ -- Author and Encode, http://www.ivast.com/products/studiosdk.html

[8] Khotake, N., Rekimoto, J., and Anzai, Y. InfoPoint: A Direct-Manipulation Device for Inter-Appliance Computing, *http://www.csl.sony.co.jp/person/rekimoto/iac/*

[9] Mississippi ETV Interactive Video Network. *http://www.etv.state.ms.us/inter-net/home.html*

[10] Liu, Q., Kimber, D., Foote, J., and Liao, C.Y. MULTICHANNEL VIDEO/AUDIO ACQUISITION FOR IMMERSIVE CONFERENCING, *Proceedings of IEEE International Conference on Multimedia and Expo*, Baltimore, MD, U.S.A., July 6-9, 2003, to appear.

[11] Liu, Q., Kimber, D., Foote, J., Wilcox, L. and Boreczky, J. FLYSPEC: A Multi-User Video Camera System with Hybrid Human and Automatic Control, *Proceedings of ACM Multimedia 2002*, pp. 484-492, Juan-les-Pins, France, December 1-6, 2002.

[12] Stenzler, M.K., and Eckert, R.R. Interactive Video, *SIGCHI bulletin*, vol. 28, no.2, April 1996.

[13] Tani, M., Yamaashi, K., Tanikoshi K., Futakawa, M., and Tanifuji, S. OBJECT-ORIENTED VIDEO: INTERACTION WITH REAL-WORLD OBJECTS THROUGH LIVE VIDEO, *Proc. of ACM CHI92*, pp. 593-598, May 3-7, 1992.

[14] Tantaoui, M.A., Hua, K.A., and Sheu, S. Interaction with Broadcast Video, *Proceedings of ACM Multimedia 2002*, pp.29-38, Juan-les-Pins, France, December 1-6, 2002.

[15] VideoClix™, VideoClix Authoring Software, *http://www.videoclix.com/videoclix_main.html*

[16] Zollman, D.A., and Fuller, R.G. Teaching and Learning Physics with Interactive Video, *http://www.phys.ksu.edu/perg/dvi/pt/intvideo.html*