

Single- and Dual-User Web Browsing in the Transparent Video Facetop

David Stotts, Jason McC. Smith, and Karl Gyllstrom

Dept. of Computer Science
Univ. of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175 USA
{stotts,smithja,gyllstro}@cs.unc.edu

ABSTRACT

The Transparent Video Facetop is a novel user interface concept that supports not only single-user interactions with a PC, but also close pair collaborations, such as that found in collaborative Web browsing, in distributed pair programming and in distributed extreme programming (dPP/dXP). We recently demonstrated the *Vis-a-Vid* facetop prototype as a single-user GUI for manipulating the elements of a traditional WIMP desktop [21]. In this paper we show how the single-user transparent video facetop can be used for fingertip control of a Web browser, and how a dual-head facetop can be used for paired synchronous (collaborative) Web browsing. The facetop is not a new browser, *per se*, but rather a novel way to interact with a Web browser, as well as a novel means of providing visual interaction among users collaboratively browsing the Web.

BASIC FACETOP CONCEPTS

The transparent video facetop is a novel enhancement of the traditional WIMP user interface, so nearly ubiquitous on today's computers. In the facetop, the user sees him/her self as a "ghostly" image apparently behind the desktop, looking back at the icons and windows from the back. Instead of a traditional desktop, we see a "face" top. This self-image is used for visual feedback and communications both to the user as well as to collaborators; it is also used for desktop/application control and manipulation via a fingertip-driven "virtual mouse".



Figure 1: Facetop physical setup, with iBot video camera

Figure 1 shows the physical setup for a computer with a facetop being displayed on a monitor. Note the video camera sitting on top the LCD panel pointing back at the user; in our current work we use a \$100 Sony iBot, giving us an image that is 640 x 480 pixels of 24-bit color, captured 30 frames per second. The facetop video window shows the PC user sitting at his/her workspace; we reverse the image horizontally so that when the user moves a hand, say, to the left, the image of the hand mirrors this movement on the screen. In software, and using a high-performance 3D-graphics video card, we make the video window semi-transparent and composite it with the desktop image itself.

Once we have the full screen video with transparent image compositing we get the illusion of the user watching the desktop from behind. Mirroring means if the user physically points to an icon on the desktop, the facetop image points to the icon as well (with proper spatial calibration of the camera and user locations). Using image analysis techniques we then track the user's fingertip in the backing window, and optionally drive the mouse from this tracker. The user can then manipulate the desktop of a *projected* computer from his seat while successfully communicating the areas of interest on the screen to others watching the projection.

Projected displays vs. monitors

The facetop as a concept works fine on a PC with any display technology -- a monitor, a projector, an immersive device -- but its unique aspects are most pronounced and most effective in a projected environment. The concept of background user video as visual cues for control and communication came about when our research group was discussing other work using a projected PC. We were all sitting in chairs viewing the projection wall, but constantly pointing at the desktop image 5 to 10 feet away. Determining where on the wall/screen to look was largely an exercise in visually interpolating along the line formed by a person's pointing arm. With the facetop, the user is right there *in* the desktop image and gives an immediate visual cue as to where to look when pointing. Figure 6 shows the facetop projected.

SYSTEM ARCHITECTURE

Our single-user facetop, shown in figures 1 through 5, is implemented on a Macintosh platform. Our collaborative facetop, shown in figure 6, is also Mac-based but runs on a peer-to-peer gigabit network between two machines, to get the very high bandwidth we need for video stream exchange. Current experimental versions are being built for best-effort use of the Internet.

The advantages of a Macintosh implementation are that the desktop is rendered in OpenGL, making its image and contents not private data structures of the OS, but rather available to all applications for manipulation or enhancement.

Though we have been speaking of the facetop as giving the user an illusion of being “behind” everything, the facetop is actually the topmost application window on the Mac desktop. It also is sized full screen, so it effectively covers the entire desktop.



Figure 2: FaceTop finger tracking (low transparency)

We dynamically control the transparency level of the facetop window, altering it from fully opaque to fully transparent during execution for varying useful effects. We can completely mask the desktop by making the facetop window fully opaque, as in figure 2. Note how the facetop window even covers and masks the title bar of the Mac desktop. A fully opaque facetop is purely a communication tool, and is especially useful in the two-head version (see figure 6) for allowing collaborators to speak face-to-face about a task without application window clutter.

We can similarly set the facetop window to full transparency; in this form, the desktop under it shows through fully and little to no user

video is visible, giving a traditional desktop appearance. Figure 3 shows a nearly transparent facetop; the only difference between this view and that of figure 2 is the transparency setting. The Web browser is running and “displayed” in figure 2 as well, but it is masked by the opaque facetop setting. If you look closely the video image of the user is very faintly visible along with the fully visible Web browser window. The near opaqueness of the browser, and the faintness of the user’s face give the illusion of the browser being “over” the video image.

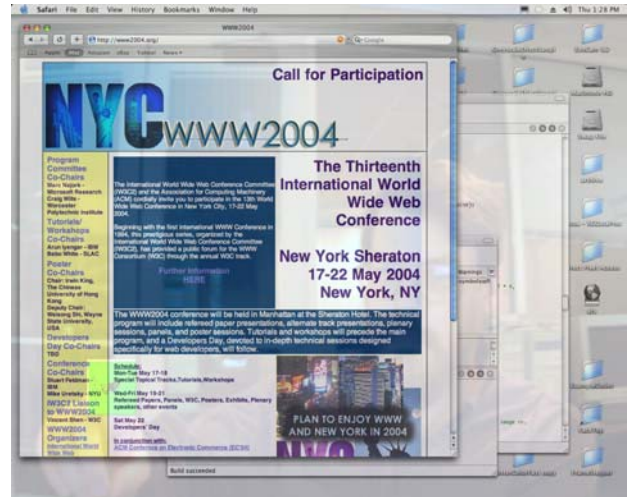


Figure 3: High transparency, mostly desktop showing

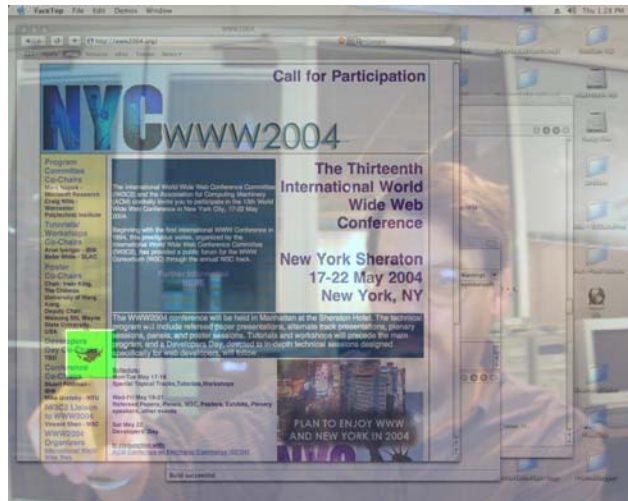


Figure 4: Mid-transparency, mix of desktop and user

Most uses for the facetop will involve a semi-transparent facetop setting, giving a mix of user video image and desktop application window content on the screen. Figure 4 shows the same desktop configuration as in figures 2 and 3, but with the facetop set to mid-transparency, making the user’s image a bit stronger. In this mix, the user’s finger can clearly be seen pointing at various hyperlinks in the browser page contents.

Implementation details

Our implementation is beautifully simple, and potentially ubiquitous due to its modest equipment needs. Facetop uses a \$100 camera, and runs with excellent efficiency on an Apple Powerbook, even when processing 30 video frames a second. No supplemental electronics are needed for wearing on the hand or head for tracking or gesture detection. Facetop is minimally invasive on the user's normal mode computer use.

The current prototype was generated with a Macintosh G4 with a high-end graphics card to perform the image transparency. We designed for the Apple Mac platform because it has better integration and access to the OpenGL layer in which the desktop is rendered. It is implemented on MacOS X 10.2 by taking advantage of the standard Quartz Extreme rendering and composition engine. QE renders every window as a traditional 2D bitmap, but then converts these to OpenGL textures. By handing these textures to a standard 3D graphics card, it allows the highly optimized hardware in the 3D pipeline to handle the compositing of the images with varying transparency, resulting in extremely high frame rates for any type of image data, including video blended with the user interface.

The video signal is generated from a Sony iBot camera, at 640 x 480 pixels in 24-bit color with no compression, at 30 frames per second. The iBot is set up 2 feet from the user at the keyboard, pointing back at the user. Note that the VAV facetop will work in principle from stored video as well as live. Tracking is done on the video frames no matter how they are generated, and signals will go to the mouse driver irrespective of video source.

The video application, with tracking capabilities, is run in a standard MacOS window, set to full screen size. Using OpenGL, setting the alpha channel level of the window to something under 0.5 (near-transparency) gives the faint user image we need.

Figure 7 shows a block diagram of our software architecture. The video flow through *Vis-a-vid* begins at the top of the diagram, from either live video capture or archived video files, represented by QuickTime Digital Video (QT DV) and QT Movies, respectively. QuickTime handles both forms in a unified manner; this allows us to intercept the video stream for analysis through the *Effects* layer. We integrate our *TrackerLib* object tracking algorithms from the OvalTine project [11,12] into a QuickTime *Effect* for simple deployment and development. OvalTine/Ovid is a system that does face-tracking in real-time video for embedding hyperlink anchors.

In the *TrackerLib*, we can use various analysis techniques to extract positions of objects in the video frame. We use this to find the coordinates of a user's fingertips, which is passed to the user interface system of *MacOS X* by having the *TrackerLib* pose as a human interface device. This allows *Vis-a-vid* to act like any other input device such as a mouse. Gesture based user events are handled by a simple plug-in to the *HIDDevice* layer that is already in the public domain. The *HIDDevice* layer then passes the interface events through to the Aqua UI layer.

Also in *TrackerLib*, we can use positional and object boundary information to alter the video stream for visual feedback to the user. Since we are integrated with the QuickTime Effects system, we can use the various real-time filters and effects to perform our manipulations. One such possibility is the use of QuickTime Sprites for marking up archived content. Sprites are an animated overlay layer that can be used for per-object visual tracking

feedback. Another is the use of the Edge Detection filter to create a minimally intrusive line-drawing effect for the feedback video.

The transparency is handled by the *Quartz Extreme* layer, which combines the video and UI streams into a series of *OpenGL* textures with appropriate alpha channels, which are then composited by the accelerated hardware's 3D *OpenGL* pipeline and sent to the screen.

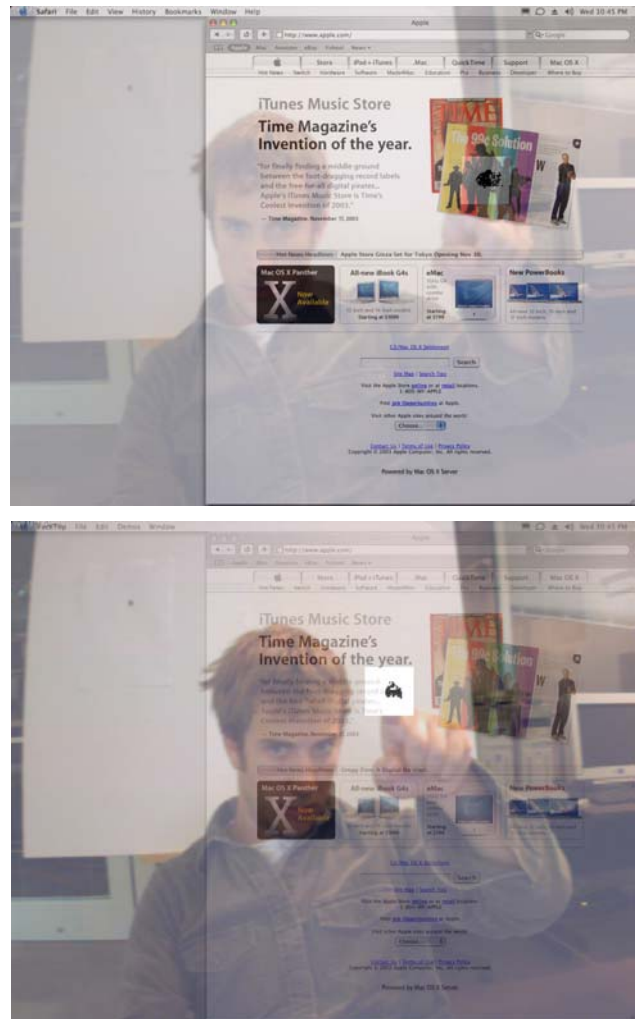


Figure 5: Mousing and clicking with fingertip gestures

WEB BROWSING IN FACETOP

A major application area for the facetop is in Web browsing, and the main facetop feature we exploit to do so is finger tracking. Figure 1 illustrates the tracking in a view of the facetop when it is fully opaque, showing the user and none of the underlying desktop. The highlighted box around the finger is the region the tracker operates in, and in this view we show the actual data bits being examined (a debugging mode that can be toggled on and off). As the user moved the hand around in view of the camera,

the tracker constantly finds the center of mass off the fingertip and reports an $\langle x,y \rangle$ coordinate location for each frame.

We presume for this discussion that one has a Web browser that supports mouse gestures. Opera is one browser that does this (<http://www.opera.com>); another is Safari on the Macintosh, after the addition of the freely downloadable Cocoa Gestures plug-in package. Our experiments so far have been with Safari.

In the facetop, the user's fingertip functions as a mouse driver, so the browser can be controlled with finger motions rather than the mouse. The tracker provides the $\langle x,y \rangle$ location information for moving the mouse; the more difficult problem is designing and implementing gestures that can serve as mouse clicks, drags, etc.

The current VAV facetop implementation has several other useful features, most activated by key presses that act as on/off toggles. User image transparency, for example, is altered from faint to opaque with the left and right arrow keys. The facetop always internally tracks the user fingertip, but moving the mouse pointer during tracking can be toggled on and off. The search neighborhood can be viewed as a box on the screen at the fingertip (see figure 1 for example); this mode shows in the box the filtered bits that the tracker actually works with, rather than showing the source image.

Link activation in the browser

As in mouse-based browsing, a link in a Web page is followed when the mouse is clicked on it. The facetop tracker gives us mouse-pointer location and causes mouse motion, but the harder issue is how to click the mouse. One method we use is to have a second tracker thread running, watching the lower left (and lower right) 150 x 150 pixel corner of the screen. When a browsing user wants to "click" the mouse (mouse down) a finger is raised in the corner. When the finger is removed, a mouse up event is registered. Thus a mouse "click" (mouse up then mouse down) is done by raising and lowering the non-browsing finger in the screen corner, while the browsing finger is holding the mouse pointer on a link anchor.

Another mode we have for mouse click activation is key presses. Facetop recognizes the down arrow key as a toggle between "mouse up" and "mouse down". Thus a mouse click is done by finger tracking the mouse to a link, then a double press on the down arrow key (we also allow the "z" key for use of the opposite hand). A mouse drag is a press on the down arrow key, track/move the mouse, then another down arrow key press (the ending mouse up).

Since finger pointing is subject to some jitter, both from finger shaking and from slight tracker variability from frame to frame, we use a "neighborhood search" when clicking on a Web link. Most Web pages have link anchors that are much wider than tall (lines of text); we presume, then, that the user has better horizontal latitude than vertical and thus search up and down from the current mouse position for the link to follow. Once a mouse click is activated, the facetop gets the current coordinates for the pointer from the tracker and queries the browser a few pixels in either vertical direction from there for an active link. Once an active link anchor is found, the mouse click event is sent to the browser at the link location.

Finger gestures for more browser control

In addition to mouse movement and clicking via finger movements, we have trained the facetop with several mouse gestures for other

browser controls, using the Cocoa Gestures package for Safari (Cocoa Gestures allows adding mouse gestures to any Mac application written for the Cocoa API). For example, when a user turns on finger tracking and wipes the finger to the left, this activates the browser "back" function for the history list. Similarly a finger wipe to the right activates the "forward" function on the history list.

These finger gestures are analogous to mouse gestures, in that they are only in effect if the finger is wiped when the "mouse down" event is in force and the "mouse up" event has not happened. This mirrors the event chain when a mouse is clicked and held, then dragged right or left, then released. The movements between mouse down and mouse up are interpreted as the encoded action.

Figure 5 illustrates the finger wiping "back" gesture during browsing. It also illustrates a potential cognitive loading problem in the facetop. The earlier screen shots (1 through 4) showed a user with a visually busy, cluttered background. Most facetop applications will be enhanced, and the potential visual confusion reduced, by the user sitting against a neutral colored, plain background, more like the one in figures 5 and 6. We are experimenting with different image rendering techniques as well for reducing visual confusion in browsing. Instead of showing the user in realistic video, for example, the same visual cues might be given by showing a gray-scale, embossed image. We would want to switch back to realistic video when the facetop was made opaque for use as a communication tool (during collaborative browsing, as in the next section).

PAIRED WEB BROWSING

An equally interesting domain of application for the facetop is in collaborative systems – specifically in systems for supporting synchronous paired tasks. We have been investigating such a system for the past year for use in distributed Pair Programming and distributed Extreme Programming (dPP/dXP) [1,2]. Pair programming is a software engineering technique where two programmers sit at one PC to develop code. One types ("drives") while the other reviews and assists ("navigates"); roles swap frequently. The benefits of pair programming are well known in co-located situations [3]; we have been exploring if they remain in distributed contexts [9,10,20].

The facetop was developed in the context of this synchronous close collaborative work. We have extended our experiments to include its use in paired collaborative Web browsing. For paired Web browsing, the primary advantage the facetop gives over other approaches is the close coupling of communications capabilities with examination of the content. Each user can see where the other points in the shared Web page; they can also use the facetop as a direct video conferencing tool without changing applications or interrupting the Web-related activities.

For the dual-user facetop, we have built a setup that has both video streams (each collaborator) superimposed on a shared desktop, illustrated in Figure 6. Our current prototype uses the VNC system for desktop sharing (<http://www.realvnc.com/>). Each user sits slightly to the right so that the two heads are on different sides of the frame when the two streams are composited. In this "knitted together" joint image, we sit each user against a neutral

background to control the possible added visual confusion of the dual facetop image.

We are working on a custom modification of VNC that will pass uncompressed data. In our current facetop we have real-time smooth video but “chunky” screen/desktop updates. This works for web browsing but needs to be better for more rapidly changing applications. Collaborating users also communicate audibly while using the facetop via an Internet chat tool like Yahoo messenger. We have not built audio services into the facetop itself, and see no need to do so given the external availability of these capabilities in several forms.

As noted earlier, we originally built this paired facetop to use in our distributed pair programming research. These experiments have been underway for 2 years in non-video environments, and we know from them that programmers are successfully constructing software systems via remote synchronous collaborations. Collaborative Web browsing is a less complex task and so is equally well-suited for use as a facetop application.

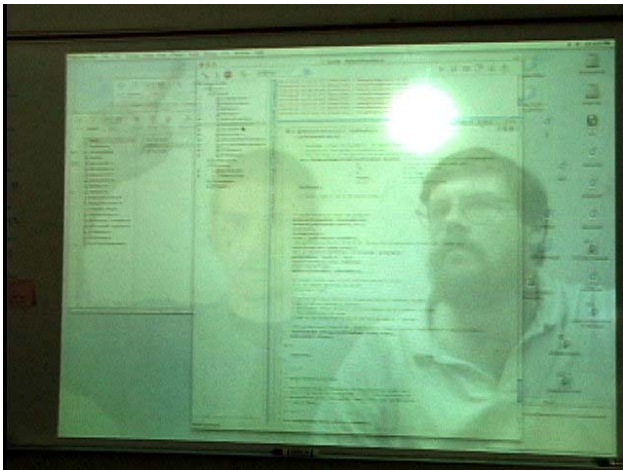


Figure 6: Dual-head FaceTop for collaborative browsing

Chalk passing

Passing locus of control among collaborators in a shared application is an important issue, called *floor control*, or *chalk passing*. The user who has “the chalk” is the one who drives the mouse and click on links when Web browsing.

Our tracker algorithm has a loss recovery mode that produces an interesting chalk passing behavior in the dual-user facetop. When tracking, if the user moves the finger faster than the tracker can track, we detect that it is “lost” by noticing no data for processing in several consecutive frames. When this happens, the algorithm stops tracking in a local neighborhood and does an entire image scan; this is too computationally expensive to do each frame, but works well for the occasional frame. In this full-frame search, the tracker acquires and moves to the largest fingertip object it finds.

With two users, this means that chalk passing happens simply by the user with the mouse hiding (dropping, moving offscreen) the finger. This “loses” the tracker and starts the full screen search algorithm. The mouse pointer immediately jumps to the other user’s fingertip

and so control passes. If there is no finger to track the tracker “parks” in a corner until there is one.

An issue to be furthered examined is if the facetop gives the two users a better sense of presence, or better techniques for communicating ideas -- visually, drawing, and gestures. Our dPP experiments continue to show that collaborating programmers want the ability to point at the work they are sharing, to save time otherwise spent in verbal descriptions and instructions [9,20]. Users collaboratively browsing the Web have the same needs and desires. The facetop gives them a new, additional capability, which is to point at Web page content without having to receive control of the mouse. The user in control of the mouse can continue to type, fill out Web forms, or otherwise navigate the browser window while the other user points and discusses via the video stream.

RELATED PRIOR WORK

The facetop combines and extends work from several different domains of computing research. Gesture-based computer controls have existed for a while, for example. The facetop, however, is unique among these for two reasons. The first is transparency: the facetop blends the traditional desktop with a video stream of the user, mirrored and made semi-transparent. The second is the video cues the user image gives: the user is *in* the desktop, as live background wallpaper, rather than making detached gestures apart from the image of the desktop. These video cues have proven very effective at giving fine and intuitive control of the mouse cursor to the user in various tasks and applications we have experimented with.

Transparency, UI, Video, and Gestures

Many prior research projects have experimented with aspects of what we have unified in the facetop. Several researchers have made systems that have transparent tools, windows, pop-ups, sliders, widgets that allow see-thru access to information below; these are primarily used for program interface components [13,15]. Many systems have some user embodiment and representation in them (avatars), especially in distributed virtual environments like [14], but these tend to be generated graphics and not live video. Giving your PC “eyes” is a growing concept, as is illustrated by this 2001 seminar at MIT [16]. A system being developed in Japan [17] uses hand activities as signals to programs; the system uses silhouettes to make recognition easier and faster. Our ideas for fingertip gesture control in the facetop are related to the many efforts under way to recognize pen gestures and other ink-based applications; the TabletPC based on Windows with ink is now commercially available from several manufacturers. They are also related to efforts in the past to recognize human facial features and motions.

Hand-based user input devices are available, like the P5 glove from Essential Reality (see P5 features on the company website at http://www.essentialreality.com/p5_glove.asp). A glove user wears a sensor net on the hand and the input from the device is used to determine hand motion and gesturing, allowing mouse driving as well as other virtual environment control activities. In the facetop, we do the gestures from video analysis alone.

The work most closely related to our facetop video analysis is from the image-processing lab of Tony Lindberg in Sweden. Researchers there have developed tracking algorithms for capturing hand motions rapidly via camera input, and have developed demonstrations of using tracked hand motions to interact with a PC [18,19]. One application shows a user turning on lights, changing TV channels, and opening a PC application using various hand gestures while seated in front of a PC. Another experiment shows careful tracking of a hand as it displays one, two, and three fingers, and scales larger and smaller. A third experiment uses hand gestures in front of a camera to drive the mouse cursor in a paint program.

The missing concept in Lindberg's work (and in other hand-gesture work), one that we are exploiting for *Vis-a-vid*, is the immersion of the user *into* the PC environment. In Lindberg's work the user is still an object separate and apart from the PC being interacted with. In the facetop, the user is given the illusion of being part of the environment being manipulated. We think this immersion gives very useful and important visual cues that are absent in earlier gesture experiments. These visual cues give the feedback needed by a user to fine-grained control of the desktop, and also give a more naturally learned and manipulated interface. We are currently testing these hypotheses.

Collaborative systems, distributed workgroups

One major use for the facetop is in collaborative systems. There have been far too many systems built for graphical support of collaboration to list in this short paper. Most have concentrated on synthetic, generated graphics. *ClearBoard* [4] is one system that is especially applicable to our research. Clearboard was a non-co-located collaboration support system that allowed two users to appear to sit face to face, and see the shared work between them. The ClearBoard experiments showed that face-to-face visibility was enhancing to collaboration effectiveness. However, the workstations required were expensive and used custom-built hardware. One of the advantageous points of the facetop is its use of cheap and ubiquitous equipment.

We are also leveraging the results of some wall-size display experiments at UNC [6, 7]. Whimsically termed the "*Office of Real Soon Now*" (a play on the name of the "*Office of the Future*" [5]), it aims to get some of the benefits of large screens without waiting years and spending large amounts of money. In this project, Bishop and Welch have produced double-width wall-sized displays for their offices using COTS projectors, video cards, and PCs. For their experiments they completely abandoned CRT displays and used only projected wall displays; after 4 years neither has any intention to return to CRTs. Benefits of the large wall displays include greatly reduced eyestrain; better interaction capabilities with students when discussing joint work; and expanded screen real estate. Their experiments have focused on individual and co-located group use of the wall displays, and have not involved networked collaborations.

Numerous collaborative Web browsers have been built, including recently TWiki (<http://twiki.org/>), CobWeb [22], CWB [23], and CoVitesse (<http://ihtm.imag.fr/demos/CoVitesse/>). To solve the problem of allowing 2 or more users to access simultaneously the same Web pages, and allow some floor control to determine which users will be able to direct the group progression from page to page. The added advantage of the facetop over these previous efforts is

the integration of the user as video representation, along with concurrent audio and video communications.

One last project we use results from is BellCore's *VideoWindow* project [8]. In this experiment, two rooms in different buildings at BellCore (coffee lounges) were outfitted with video cameras and wall-sized projections. In essence, an image of one lounge was sent to the other and projected on the back wall, giving the illusion in each room of a double-size coffee lounge. The researchers discovered that many users found the setup to be very natural for human communication, due to its size. Two people, one in each room, would approach the wall to converse, standing a distance from the wall that approximated the distance they would stand from each other in face-to-face conversations. The conclusion:

Video, when made large, was an effective and convincing communication tool.

We leveraged this finding in creating the dual-head facetop that we use for synchronous, collaborative Web browsing.

FOR FURTHER RESEARCH

We are experimenting with the facetop in the context of Web browsing and other single-user and collaborative applications. Here are several of the questions we are investigating in these experiments:

- How effective is the VAV facetop as a mouse-replacement in a traditional WIMP desktop?
- What is the most effective camera angle and placement for comfortable arm movement and hand motion in a VAV facetop?
- How does VAV effectiveness compare in a projected environment vs. a CRT-based environment?
- How do single users rate the adoptability of the VAV facetop?
- How do distributed pairs rate the adoptability of VAV facetop?
- Do distributed pairs perform their tasks better with VAV facetop?
- Will two overlaid VAV facetop's work well as support for close synchronous paired collaboration?
- Are there uses for the VAV in multi-user environments other than paired collaborations?
- Can we use a broad range of user gestures in the VAV for desktop and program control?
- How can the VAV be technically implemented for different platforms with different levels of built-in or accelerated graphics support?

Several issues are natural to continue to investigate in the single-user facetop as well. Will users find moving the arm and hand in the air too tiring? Can we make it work if the user never has to lift his arm (adjust camera location and angle)? What gestures are simple to make and easy to recognize for common WIMP actions like clicking and dragging? What subset of WIMP desktop control will users find acceptable for facetop control vs. traditional mouse control?

The natural extension to the basic VAV concept of finger tracking

is broader gesture recognition. We want to expand the tracking capabilities to allow a broader range of user actions to be recognized and tracked, perhaps to even include head motions. We want to use the hand and gesture tracking algorithms developed by Lindberg [18,19] initially. This work allows detection of multiple fingers, hand rotations, and scaling to and from the screen.

CONCLUSIONS

The transparent video facetop is a novel user interface technology that has application in many computer application areas, including both single-user Web browsing and collaborative web browsing. The facetop works efficiently with an inexpensive firewire camera and laptop speeds, making the concept potentially ubiquitous. Manipulation of the browser controls is accomplished by finger movement and pointing, combined with mouse gesture recognition software in the browser. The facetop enhances collaborative Web browsing beyond the capabilities of previous systems in several ways. The users are able to see each other, to visually and audibly communicate and discuss content, and to point to content without having to pass the mouse control.

Acknowledgements This work was partially supported by a grant from the U.S. Environmental Protection Agency, # R82-795901-3.

REFERENCES

- [1] Beck, K., *Extreme Programming Explained*, Addison-Wesley, 2000.
- [2] Wells, J. D., "Extreme Programming: A Gentle Introduction," 2001, available on-line at <http://www.extremeprogramming.org/>
- [3] A. Cockburn and L. Williams, "The Costs and Benefits of Pair Programming," *eXtreme Programming and Flexible Processes in Software Engineering -- XP2000*, Cagliari, Sardinia, Italy, 2000.
- [4] H. Ishii, M. Kobayashi, and J. Grudin, "Integration of inter-personal space and shared workspace: ClearBoard design and experiments," *Proc. of ACM Conf. on Computer Supported Cooperative Work*, Toronto, 1992, pp. 33-42.
- [5] H. Fuchs, "The Office of the Future," pp. <http://www.cs.unc.edu/~raskar/Office/>.
- [6] G. Bishop, , pp. <http://www.cs.unc.edu/~gb/office.htm>, The Office of Real Soon Now.
- [7] G. Bishop and G. Welch, "Working in the Office of 'Real Soon Now'," *IEEE Computer Graphics and Applications*, pp. 76-78, July/August 2000.
- [8] R. S. Fish, R. E. Kraut, and B. L. Chalfonte, "The VideoWindow System in Informal Communications," *Proc. of ACM Conf. on Computer Supported Cooperative Work*, Los Angeles, 1990, pp. 1-11.
- [9] P. Baheti, L. Williams, E. Gehringer, and D. Stotts, "Exploring the Efficacy of Distributed Pair Programming," *XP Universe 2002*, Chicago, August 4-7, 2002; *Lecture Notes in Computer Science 2418* (Springer), pp. 208-220.
- [10] P. Baheti, L. Williams, E. Gehringer, D. Stotts, "Exploring Pair Programming in Distributed Object-Oriented Team Projects," *Educator's Workshop, OOPSLA 2002*, Seattle, Nov. 4-8, 2002, accepted to appear.
- [11] Smith, J., D. Stotts, and S.-U. Kum, "An Orthogonal Taxonomy for Hyperlink Anchor Generation in Video Streams using OvalTine," *Proc. of Hypertext 2000 (ACM)*, May, 2000, San Antonio, TX, pp. 11-18.
- [12] Stotts, D. and Smith, J., "Semi-Automated Hyperlink Markup for Archived Video," *Proc. of Hypertext 2002 (ACM)*, College Park, MD, May 2002, pp. 105-106.
- [13] Eric A. Bier, Ken Fishkin, Ken Pier, Maureen C. Stone, "A Taxonomy of See-Through Tools: The Video, Xerox PARC, Proc. of CHI '95," <http://www.acm.org/sigchi/chi95/Electronic/documnts/videos/eab1bdy.htm>
- [14] Steve Benford, John Bowers, Lennart E. Fahlén, Chris Greenhalgh and Dave Snowdon, "User Embodiment in Collaborative Virtual Environments," Proc. of CHI '95, http://www.acm.org/sigchi/chi95/Electronic/documnts/papers/sdb_bdy.htm
- [15] Beverly L. Harrison, Hiroshi Ishii, Kim J. Vicente, and William A. S. Buxton, "Transparent Layered User Interfaces: An Evaluation of a Display Design to Enhance Focused and Divided Attention," Proc. of CHI '95, http://www.acm.org/sigchi/chi95/Electronic/documnts/papers/blh_bdy.htm
- [16] Vision Interface Seminar, Fall 2001, MIT, <http://www.ai.mit.edu/~trevor/6.892/>
- [17] T. Nishi, Y. Sato, H. Koike, "SnapLink: Interactive Object Registration and Recognition for Augmented Desk Interface," Proc. of IFIP Conf. on HCI (Interact 2001), pp. 240-246, July 2001.
- [18] Bretzner, L., and T. Lindberg, "Use Your Hand as a 3-D Mouse, or, Relative Orientation from Extended Sequences of Sparse Point and Line Correspondences Using the Affine Trifocal Tensor," Proc. of the 5th European Conf. on Computer Vision, (*H. Burkhardt and B. Neumann, eds.*), vol. 1406 of *Lecture Notes in Computer Science*, (Freiburg, Germany), pp. 141--157, Springer Verlag, Berlin, June 1998.
- [19] Laptev, I., and T. Lindberg, "Tracking of multi-state hand models using particle filtering and a hierarchy of multi-scale image features," Proc. of the IEEE Workshop on Scale-space and Morphology, Vancouver, Canada, in *Springer-Verlag LNCS 2106* (M. kerckhove, ed.), July 2001, pp. 63-74.
- [20] Stotts, D., L. Williams, et al., "Virtual Teaming: Experiments and Experiences with Distributed Pair Programming," TR03-003, Dept. of Computer Science, Univ. of North Carolina at Chapel Hill, March 1, 2003.
- [21] Stotts, D., J. McC. Smith, and D. Jen, "The Vis-a-Vid Transparent Video FaceTop," *UIST '03*, Vancouver, Nov. 3-6, 2004, pp. 57-58.
- [22] Stotts, D., S. Kim, J. Navon, J. Prins, and L. Nyland, "CobWeb: Visual Design of Collaboration Protocols for Dynamic Group Web Browsing," *Visual Computing 2002 (Distributed Multimedia 2002)*, San Francisco, Sept. 26-28, 2002, pp. 595-598.
- [23] Esenther, A.W., "Instant Co-Browsing: Lightweight Real-time Collaborative Web Browsing," Proc. of WWW 2002, <http://www2002.org/CDROM/poster/86/>.

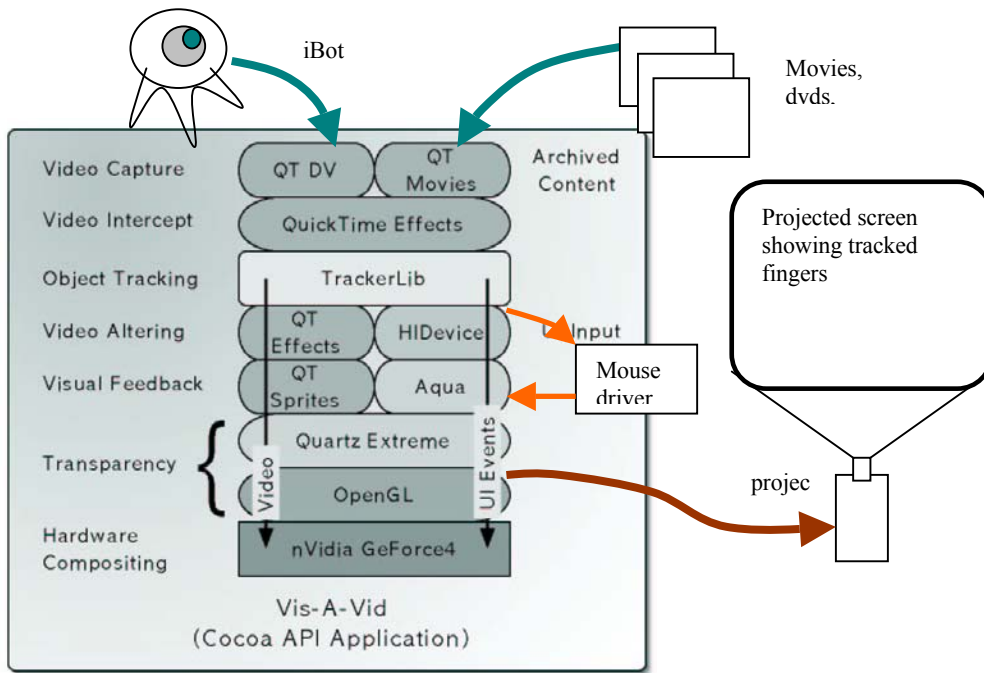


Figure 7: Vis-a-vid facetop system block diagram