

# Support for Distributed Pair Programming 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, remote medicine, and in distributed pair programming. In this paper we discuss the use of a novel video-based UI called the Facetop [16] for solving several problems reported to us by teams doing distributed pair programming. Specifically, the Facetop allows a distributed pair to recapture some the facial expressions and face-to-face communications contact lost in earlier distributed sessions. It also allows members of a distributed pair to point conveniently, quickly, and naturally to their shared work, in the same manner (manually) that they do when seated side-by-side. Our results enhance the ability of organizations to do effective XP-style agile development with distributed teams.

## 1 Distributed Pair Programming

Previous research [17,19] has indicated that pair programming is better than individual programming in a co-located environment. Do these results also apply to distributed pairs? It has been established that distance matters [18]; face-to-face pair programmers will most likely outperform distributed pair programmers in terms of sheer productivity. However, the inevitability of distributed work in industry and education calls for research in determining how to make this type of work most effective. Additionally, Extreme Programming (XP) [1,2] usually has co-located pairs working in front of the same workstation, a limitation that ostensibly hinders use of XP for distributed development of software.

We have been investigating a video-enhanced programming environment 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 [6,7,15].

Video was one issue discussed at a workshop on distributed pair programming at XP/AU 2002. This workshop was attended by over 30 people, many of whom had tried some form of distributed pair programming and were working on tools to improve the effectiveness of such activities. The consensus on video was that “web cam” style video – small image and low frame rate – was of little value in enhancing communications or sense of presence in a distributed pairing. However, it was felt that video, if large enough and real enough was of potential value and worth further research. We have been doing that research since that time.

## 2 The Facetop Basics

The transparent video Facetop [16] 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".

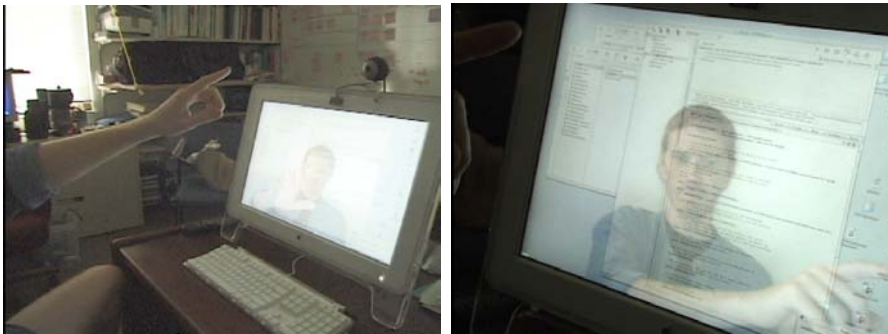


Figure 1: Facetop physical setup, with iBot video camera

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. Figure 2 shows this finger tracking (the desktop image is more transparent and the user face more opaque to emphasize the tracking). The user can then manipulate the desktop of a *projected* computer, for example, from his seat while successfully communicating the areas of interest on the screen to others watching the projection.

### 1.1 Transparency combined with user self-view

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 cursor to the user in various tasks and applications we have experimented with.

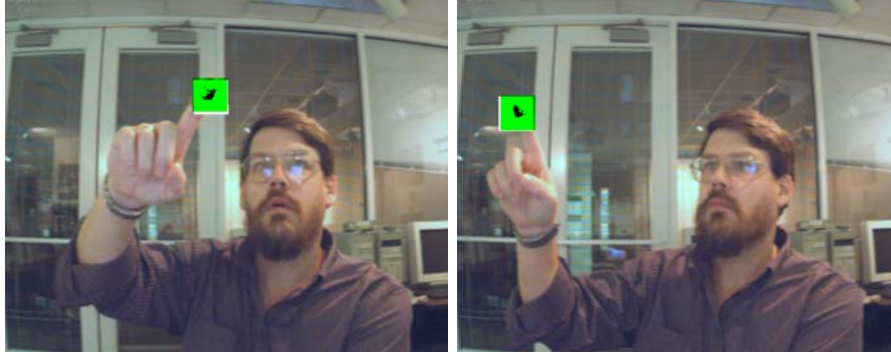


Figure 2: Facetop finger tracking (low user transparency)

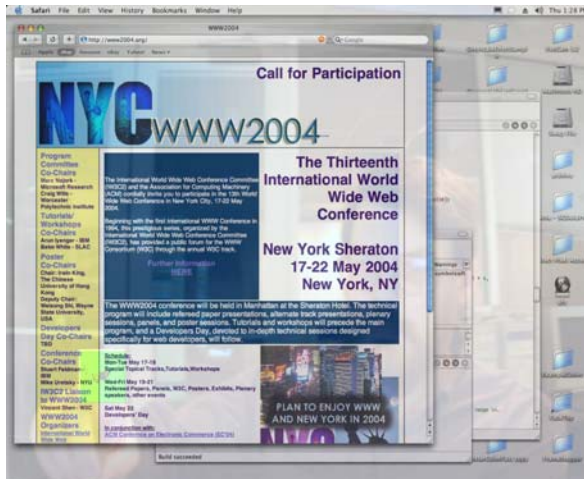


Figure 3: Varying user transparency, from mostly user showing to mostly desktop showing

We allow the user to dynamically control the transparency level of the Facetop window, altering it from fully opaque (all user face, a communications tool) to fully transparent (all desktop) during execution for varying useful effects. Figure 3 shows the near extremes.

### 3 Dual-head Collaborative Facetop

Though the previous presentation has been in the context of a single-user PC interface, 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 a two-head Facetop for the past year for use in distributed Pair Programming (dPP). This investigation is an extension of earlier studies we conducted to see if distributed pairs could pair program effectively communicating over the Internet [6,7,15].

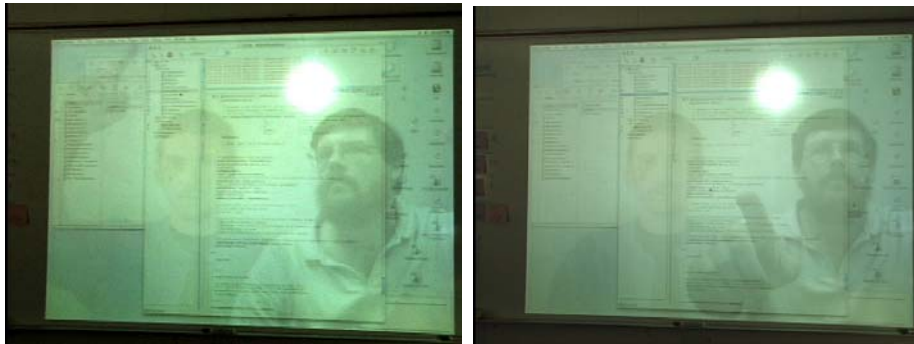


Figure 4: Dual-head Facetop for collaborative browsing

In our previous dPP experiments, programmers worked as a pair using COTS software, including *pcAnywhere* (Symantec) and *Yahoo messenger* (for voice communications). The *pcAnywhere* shared desktop allows the two programmers effectively to work on a single host computer; each seeing exactly what the other sees, as they would sitting side-by-side at the host PC. Our experiments found that programmers working in this dPP environment were as effective as co-located pairs. In post-trial interviews, teams consistently told us 3 things:

- They missed facial expressions and the sense of presence
- They wanted a way to point at the shared work they were discussing via audio
- They wanted a whiteboard for drawing and design work

The Facetop provides potential solutions to each of these problems, via its video capabilities. Video was provided to the pairs in our previous dPP experiments; we gave each team “web cams” that generate small images at low frame rates. Each team turned off the video almost immediately, finding that the small, nearly still, images gave no useful information, but did consume considerable bandwidth. Maximal bandwidth was needed for fast update of the *pcAnywhere* shared desktop.

The video capabilities in Facetop are very different, however. The image is large, and frame rates run from 15 to 30 fps, showing facial details and fine motor movements of the fingers and lips. The video image is also tightly and seamlessly integrated with the shared workspace via transparency, thereby eliminating the “dual” nature of video teleconferencing solutions. Users do not have to switch their attention from desktop, to video, back to desktop.

For the dual-user Facetop, we have built a setup that has both video streams (each collaborator) superimposed on a shared desktop, illustrated for a projected environment in Figures 4 and 5. 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.

Collaborating users continue, as before, to communicate audibly while using the Facetop via an Internet chat tool like *Yahoo messenger*. 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 workspace; they can also use the Facetop as a direct video conferencing tool (by varying the transparency level to fade the desktop image) without changing applications or interrupting the work activities.

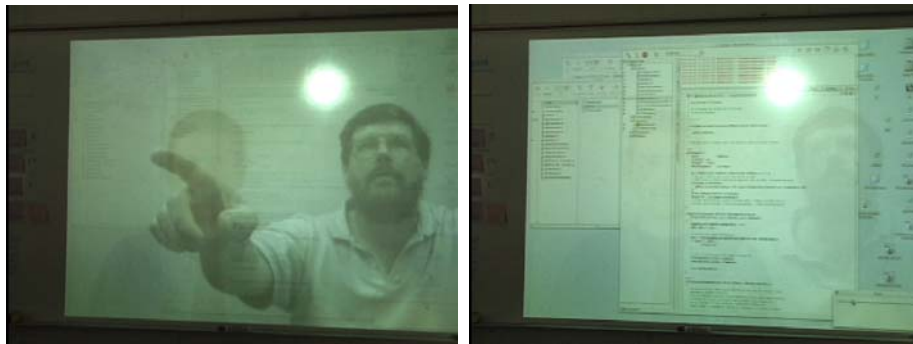


Figure 5: Varying levels of transparency in dual-head Facetop

### 3.1 System Features and Functions

The following sections briefly discuss a collection of features and functions of our current Facetop implementation.

**Multiple varying transparency levels.** In the dual-head Facetop, each user has transparency level controls that are independent of the settings chosen by the partner. A user can set the level (from opaque to transparent) of each video image separately (self and partner image), as well as level of the desktop (see Figure 5). In this way, each user can get different communications effects. If both user images are set to *highly visible*, and the desktop set *low*, the Facetop is a form of video conferencing system. Bring the desktop up to *visible* and the unique integration of user image with shared work happens, allowing pointing and discussion. Some users may wish not to see themselves and have only the partner image visible on the desktop; they can still effectively point by finger tracking and watching the mouse pointer.

**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 off screen) 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 "parks" in a corner until there is one.

**Monitor or projector.** 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. When projected, it is natural to point with hand and finger at the projected image on a wall, especially when several people in a room are viewing the projection.

**Finger tracking on/off.** One interesting feature in the Facetop is finger tracking. This function can be turned on or off and used as needed. Even if the user chooses not to use finger tracking, the Facetop has great value as a pure communication tool via finger pointing and facial expressions, especially in collaborative applications like dPP. However, tracking and mouse control does add some interesting and useful capabilities for users that wish to use them.

Figure 2 illustrates the tracking in a view of the Facetop when the user is fully opaque, showing the user and none of the underlying desktop or whiteboard. 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.

In the Facetop, the user's fingertip functions as a mouse driver, so applications like browsers 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.

**Fingertip mouse click activation.** The Facetop tracker gives us mouse-pointer location and causes mouse motion, but the harder issue is how to click the mouse. The method we currently use is occlusion of the fingertip. When the mouse pointer has been positioned, the user makes a pinching fist of sorts, hiding the fingertip in the hand or between the other fingertips. The tracker notes the loss of the tip, and begins a timer. If the tip reappears (user raises the finger) in a  $\frac{1}{2}$  second, a single-click mouse event is generated at the mouse pointer location. If the tip remains hidden for between  $\frac{1}{2}$  and 1 second, a double-click event is generated. User studies (discussed in a later section) have so far shown that this motion is not hard to learn and even master. It is sufficient to open/close windows, drag them, resize them, select links in Web browsers, and even position the mouse between characters in documents.

Another interaction method we have implemented is voice commands. This is especially useful in rapidly altering the transparency level of the various Facetop camera images, as well as for hands-free mouse clicking where useful.

**Video auto on/off.** Another technique we use for managing visual clutter is to have the Facetop tracker recognize when the fingertip enters the video frame. When the fingertip enters, the user camera image is composited in. When the tip leaves, the user fades and the desktop remains. This is modal and can be turned on and off. It is especially useful for doing presentations in Web browsers and PowerPoint.

## 4 Initial User Evaluations

Controlled user evaluations are still ongoing, but we have some usability results to report from our first experiments. To date we have had 15 users try the basic Facetop to determine if live background video is a viable, usable concept as an interface for manipulating the PC environment. We set up the Facetop up in a room with white walls so that there would not be a busy background to add visual clutter to the screen image.

As might be expected, arm fatigue is a problem for continuous use of the fingertip-based mouse feature. For browsing a hypertext, this is not a major issue, as much time is spent reading vs. actually manipulating the screen. Users drop their arm during these quiescent periods, and then raise it to point when ready to navigate more. The video image on-screen gives the visual cues needed for nearly instant positioning of the mouse pointer directly where needed.

Another problem reported by several users is visual clutter. Most users adapted quickly and comfortably to the moving image as background “wallpaper”; transparency was set at different levels by different users, and there did not seem to be a preferred level of mixing of desktop with user-image other than to say that both were visible. The human eye/brain is able to pay attention (or ignore) the face or the desktop respectively, depending on the cognitive task – depending on whether the user wants to read the screen contents or to communicate (in the two-head version).

Users were queried specifically as to visual clutter or confusion. A few objected, but most found the adjustability of transparency fine-grained enough to get to a level where they were not distracted or hindered in using the desktop.

We also created a networked tic-tac-toe game for usability trials of the dual head version and had 11 pairs of users try it. The users were a class of 8-grade students who came to the department for research demonstrations. Five of the users took less than 5 minutes to become facile with the interface, learning to move and click the mouse well enough to Web browse. All users were able to successfully play the game (which involves clicking on GUI buttons) in the 30 minute time-frame of the trials.

#### **4.1 Distributed Pair Programming Trials**

We had five of the pairs involved in past dPP experiments (with audio and shared desktop only) try the Facetop environment for small pair programming “shakedown” tasks. Since all had tried the earlier environments, the trials were designed to see if the “video made large” features in Facetop overcame the lack of pointing ability and lack of facial expressions reported by these teams before (the lack of whiteboard they reported is still being investigated, and is discussed in the next section).

All teams were quite comfortable using the Facetop, and did not consider visual complexity or clutter an issue. We suspect this is due to concentration on programming focusing the attention on the various text windows of the desktop. All dPP teams were able to complete small programs with no problems.

They also reported setting varying levels of user image transparency to suit personal taste. Given that the video images can be completely faded out, leaving nothing but desktop, the current Facetop is “no worse” than our previous audio-only environments. However, no teams chose to completely fade out the video and use audio only. All teams left the user images visible to some extent and did use the video to point to code being discussed.

In post-trial interviews, the overall impression was that Facetop was an interesting improvement over the audio-only dPP environment used before. Each team was asked “if you were to do a longer dPP development, would you prefer to use Facetop or the original audio-only environment?” All teams expressed a preference for Facetop.

These simple usability trials do not reveal if the preference for Facetop was emotional or qualitative only, or if the added video and sense of presence increases programmer effectiveness. We find these early usability trials compelling enough, though, to start larger, controlled experiments to see if Facetop can have an impact on quantitative aspects of software, such as design quality or error counts.

## **5 Further Distributed Pair Programming Investigations**

Our studies have found that adding large, fast video via the Facetop to a dPP environment enhances the qualitative experience of the programmers. Our investigations are continuing; we are gathering quantitative data on productivity and product quality in follow-on trials. Current work is in two areas: whiteboard support, and universal access for impaired programmers.

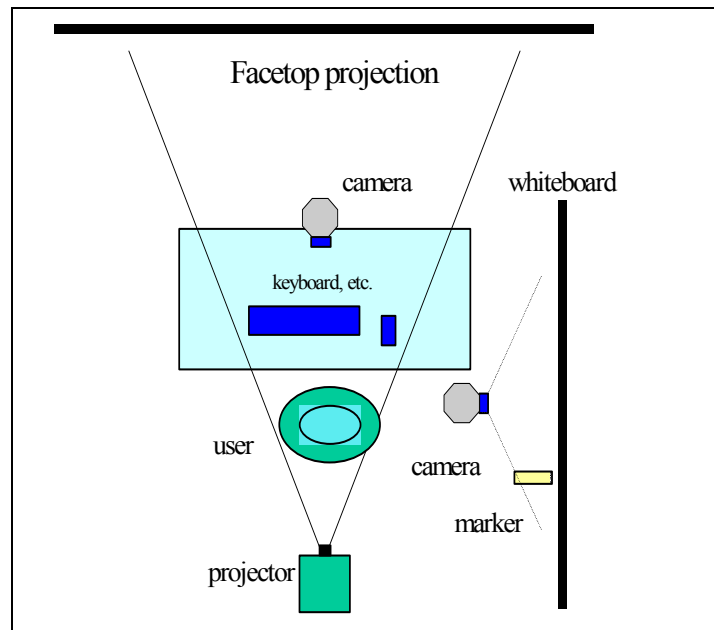


Figure 6. Schematic of two-camera Facetop for whiteboard

### 5.1 Dual camera Facetop for whiteboard

One of the items noted earlier as wanted by dPP teams in past experiments was access to a good whiteboard. To solve this problem, we have a version of Facetop that works with two Firewire video cameras per workstation. In addition to the normal Facetop user camera, a second camera is situated to the side of the user and faces a whiteboard. The user sits near enough to the board to be able to comfortably reach out from the seat and draw on the whiteboard. This layout is shown in figure 6. Facetop takes both camera streams (user face and whiteboard) and composites them into the video stream that is laid semi-transparent on the desktop. As in the normal Facetop, the user face stream is mirrored (reversed horizontally) so that pointing is meaningful to the user. The whiteboard video image is not mirrored, so that words written on the board remain readable when composited into the Facetop video.

Since the whiteboard is neutral in appearance, compositing it into the Facetop image doesn't really alter the appearance over the traditional Facetop. When words or drawings are written on the whiteboard, they appear to "float" within the room/background of the user. Figure 7 shows this compositing of both video streams. By varying transparency levels of each camera, users can see whiteboard only, or whiteboard composited with their images. Key-press commands in Facetop allow instant swapping between whiteboard image and user image. User's hands show up as drawing is done, so each sees what the other is drawing.

### 5.1 Universal access for impaired programmers

We are also investigating the use of the collaborative Facetop in providing access to pair programming, and other synchronous paired collaborations, for people with audio and visual impairments. For programmers with audio impairments, we are experimenting with the Facetop video being used for support of signing and lip reading during pair programming. Programmers with audio impairments can do side-by-side pair programming with current technology, but they cannot participate in dPP using the audio-only environments we first experimented with.

For programmers with visual impairments, we are developing audio cues that will provide information about the state of a collaboration. Currently individual programmers with visual impairments use a screen reader like JAWS [20] for navigating a PC screen. Our extensions will function similarly, but will have to not only communicate screen information, but partner activity information as well.



Figure 7. Whiteboard image composited into the Facetop user image

## 6 System Structure and Performance

Our single-user Facetop is implemented on a Macintosh platform. Our collaborative Facetop 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 30 fps video stream exchange. Current experimental versions are built for best-effort use of the switched Internet give about 18 frames a second. This is usable for dPP, but we need better for universal access and hearing-impaired signing.

A Macintosh implementation has several advantages. 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. We also use dual-processor platforms, so that one processor can handle tracking issues and other Facetop-specific loads, while leaving a processor free to support the collaborative work, such as pair programming. Video processing is handled mostly on the graphics card.

Our implementation is beautifully simple, and potentially ubiquitous due to its modest equipment needs. Facetop uses a \$100 Sony iBot 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. 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 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.

Some of our experiments have been run with the two Power Mac's connected via peer-to-peer gigabit network. In this configuration, we get a full 30 frames per second video data exchange in each direction. This is possible due to the high network speeds, and due to our passing only the 640 x 480 camera image. Image scaling to screen size is handled locally on each machine after the 2 video signals and the desktop are composited into one image.

## 7 Related Prior Research

### 7.1 Pointing in Collaborative Applications

Several systems have dealt with the issue of two users needing to provide focus (point) at different, or independent locations on a shared screen. The common solution is to provide two mouse pointers and let each user control his/her own independently. Use of two mouse pointers is central to a dPP tool being developed by Hanks [21]. This is fundamentally different from using a human device (fingers) to point as in Facetop.

### 7.2 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 enhanced the effectiveness of collaboration. However, the workstations required were expensive and used custom-built hardware. One of the advantages of the Facetop is its use of cheap and ubiquitous equipment.

One last project we use results from is BellCore's *VideoWindow* project [5]. 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 have leveraged this finding in creating the dual-head Facetop that we use for synchronous, collaborative Web browsing.

### 7.3 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 [8,11]. Many systems have some user embodiment and representation in them (avatars), especially in distributed virtual environments like [10], 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 [12]. A system being developed in Japan [9] 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 Tablet PC 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.

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 [13,14]. 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 display 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 Facetop, is the immersion of the user *into* the PC environment to give video cues and feedback for control.

**Acknowledgements.** This work was partially supported by a grant from the U.S. Environmental Protection Agency, # R82-795901-3. It does not represent the official views or opinions of the granting agency.

### 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. 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.
6. 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.
7. 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.

8. 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/eablbdy.htm>
9. 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.
10. 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](http://www.acm.org/sigchi/chi95/Electronic/documnts/papers/sdb_bdy.htm)
11. 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](http://www.acm.org/sigchi/chi95/Electronic/documnts/papers/blh_bdy.htm)
12. Vision Interface Seminar, Fall 2001, MIT, <http://www.ai.mit.edu/~trevor/6.892/>
13. 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 5<sup>th</sup> 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.
14. 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.
15. 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.
16. 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.
17. Nosek, J.T., "The Case for Collaborative Programming," *Communications of the ACM*, March 1998, pp. 105-108.
18. Olson, G.M., and J.S. Olson, "Distance Matters," *Human-Computer Interaction*, vol. 15, 2000, pp. 139-179.
19. Williams, L., "The Collaborative Software Process," *Ph.D. dissertation*, Dept. of Computer Science, Univ. of Utah, Salt Lake City, UT, 2000.
20. JAWS, Windows screen reader, Freedom Scientific, <http://www.freedomscientific.com/>
21. Hanks, B., "Distributed Pair Programming: An Empirical Study" XP/Agile Universe, Aug. 2004, Calgary, to appear.