A Distributed Online Environment for Gesture-Based Collaboration

Cristian Gadea, Bogdan Ionescu, Dan Ionescu, Shahidul Islam, Bogdan Solomon
University of Ottawa, Mgestyk Technologies
Ottawa, Ontario, Canada
Email: {cgadea, bogdan, dan, sislam, bsolomon}@ncct.uottawa.ca

***Abstract*—Gestures are seen as the new human computer interface mechanism. They can make computers and devices easier to use, such as by allowing people to share photos by moving their hands through the air. Much research has been undertaken to develop systems whereby users can effectively control computers and devices through gestures. Existing solutions have relied on inadequate hardware, requiring elaborate setups limited to the lab. Relying heavily on image processing, gesture recognition algorithms used so far are not practical or responsive enough for real-world use. Most importantly, existing solutions have lacked a software environment that allows users to perform common collaborative tasks. In this paper, a new paradigm for next-generation computer interfaces is introduced. The method presented is based on an affordable 3D camera that is easy to set up and has a flexible detection range. Our method accurately detects hand gestures from depth data, allowing them to be used to control any application or device. The paper proposes the control of windows and their content in distributed collaborative and web-based environments on which many teams cooperate to complete useful tasks, as demonstrated with examples.**

***Index Terms*—gesture-based control, web collaboration, distributed systems, virtual environments, human computer interaction, 3D depth camera**

I. Introduction

Gesture recognition and gesture-based control has gained the attention of various research groups related to image processing and artificial intelligence since the sixties [1]. More recently, as the camera and gesture processing technology has continued to evolve, hand gesture recognition technologies have received significant interest from commercial and academic institutions looking for a compelling system that allows for touch-free interaction.

Traditionally, PCs have relied on keyboards and mice as the standard input devices for interacting with user environments based on the WIMP (windows, icons, mouse, pointer) paradigm. There are many instances, however, where user input based on hand gestures can provide an improved user experience. For example, using a mouse to drag an icon from the left side of a high-resolution display to the right side can prove to be a cumbersome affair, which may even require the user to physically lift the mouse off the table and reposition it so that they can complete the full dragging operation. By using hand gestures, users can quickly swing their arm and point to the necessary location on the screen.

The drawbacks of the keyboard and mouse as input devices are particularly apparent when considering users with special accessibility requirements, public kiosks, the “10 foot” interfaces of televisions and set-top boxes, and even industry-specific scenarios such as surgeons who are unable to come in contact with touch-based devices. Touch-free gesture control can therefore complement, or even completely replace, traditional interaction modalities in many different cases.

Fig. 1. Two-Handed Gesture Control of Virtual Workspaces.

The concept of a computer interface that is entirely controlled by gestures has been popularized by many recent science fiction movies and illustrations such as Figure 1. In the 2002 movie *Minority Report*, gesture-driven displays were used by the main character to perform forensic analysis using photographic and video data. The fluid and natural interaction that such gesture control systems offer has caught the attention of researchers who have attempted to re-create this vision with varying results.

Unfortunately, most existing solutions suffer from several shortcomings that have prevented mainstream adoption of such technology. Some of the hardware that has been used for processing gestures has required users to wear obtrusive sensors and stand near multiple carefully calibrated cameras. Most cameras used so far rely on color data and are therefore sensitive to environmental factors such as dynamic backgrounds and lighting conditions. The algorithms used to determine gestures from the data returned by the hardware have been unreliable when tested on a wide variety of users and gestures have generally been limited to basic hand-tracking. Since the time needed for the computer to recognize a gesture is usually longer than the time needed to display its result, there is always a lag affecting the practical application of such interfaces. Finally, there have not been any collaborative environments that allow users to freely use gestures for completing tasks such as sharing photos with friends. We plan to address all of these problems by using a custom 3D depth-based camera that generates data ideal for processing hand gestures so that our algorithms can reliably return the number of hands, number of fingers, and angles of the fingers that the user is showing. We then propose a distributed web-based environment that allows users to use hand gestures when collaborating with other users on maps, photos and more.

The remainder of this paper is organized as follows: Section II further analyzes how the work presented in this paper differs from the current state-of-the-art. Section III then discusses the unique architecture of the proposed 3D camera and web-based system. Section IV offers a closer look at the low-level details of the 3D camera, gesture processing software and collaboartive gesture environment. The resulting gesture-based control scenarios within a distributed online environment are presented in section V. Finally, section VI reflects on the contributions of this paper and proposes topics for future research.

II. Related Work

The processing of hand gestures has been explored extensively in existing literature. Some of the earlier work by Freeman and Weissman [2] used a video camera and computer vision template matching algorithms to detect a user's hand from across a room and allow the user to control a television set. A user could show an open hand and an on-screen hand icon would appear that could be used to adjust various graphical controls, such as a volume slider. The slider was activated when the user would cover the control for a fixed amount of time. The authors discovered that users enjoyed this alternative to the physical remote control and that the feedback of the on-screen hand was effective in assisting the user. However, users found it tiring to hold their hand up for long amounts of time to activate the different controls. This user fatigue common of gesture-based interfaces has been called *gorilla arm*.

Other approaches have relied on using multiple cameras to produce a 3D image which can be used to detect and track hand motion [3][4][5]. These systems required an elaborate installation process which had to be completed carefully as calibration parameters such as the distance between the cameras was important in the triangulation algorithms used. These algorithms were also computationally expensive since a large amount of video data needed to be processed in real-time, and stereo-matching typically fails on scenes with little or no texture. Ultimately, such systems would not be useable outside of their special lab environments.

In [6], Mistry presented the SixthSense wearable gestural interface, which used a camera and projector worn on the user's chest to allow the user to zoom in on projected maps (among other activities) by the use of two-handed gestures. In order for the camera to detect the user's hand, the user had to wear brightly-colored markers on their index fingers and thumbs. The regular webcam worn by the user would also be sensitive to environmental conditions such as bright sunlight or darkness, which would make distinguising the colored markers much more difficult, if not impossible.

Wilson and Oliver [7] attempted to create a *Minority Report*-like environment that they called GWindows. The user was able to move an on-screen cursor of a Microsoft Windows desktop by pointing with their hand and using voice commands to trigger actions like "close" and "scroll" to affect the underlying application windows. They concluded that users preferred interacting with hand gestures over voice commands and that desktop environments designed for gesture interactions were worth pursuing further.

When considering collaborative online environments, several commercial and academic web-based collaboration solutions have existed for some time [8][9]. However, interaction with other users in these environments is usually limited to basic sharing of media files, rather than allowing for full real-time collaboration of entire web-based applications and their data between users on distinctly deployed domains, as our solution proposes.

The system we developed uses a custom 3D depth camera to sense the position of the user's hand and process hand gestures in any lighting. Camera systems which recover depth information from the captured scene have gained significant traction recently with at least one affordable model announced to available in stores for consumers later in 2010 (although its resolution, image quality and ability to interface with a regular PC are still relatively unknown) [10]. Our system allows the user to perform a large variety of gesture interactions within a distributed collaborative environment known as UC-IC ("you see I see"). Users are able to select icons and windows by pointing to them. To select and drag an item, the user simply hides their index finger and swings their hand to the position they wish to drag the item to. Since UC-IC is a distributed environment, multiple users can be interacting with each other using such gestures in real-time to perform actions such as sending application windows or viewing a collaborative application window. Additional gestures implemented include a hand flicking gesture to navigate between items such as photos, and a thumbs-down gesture to close an application window. The benefits on task completion efficiency offered by flick-style gestures have been shown in [11]. The system also provides a mouse cursor for each hand so that users may perform multiple actions at the same time, such as move two windows around or resize a window. UC-IC is a web-based environment accessible via a typical web browser, meaning that there is very little for users to install in terms of setup. By offering these advantages, we believe that gesture-controlled environments can finally be accessible to the average person

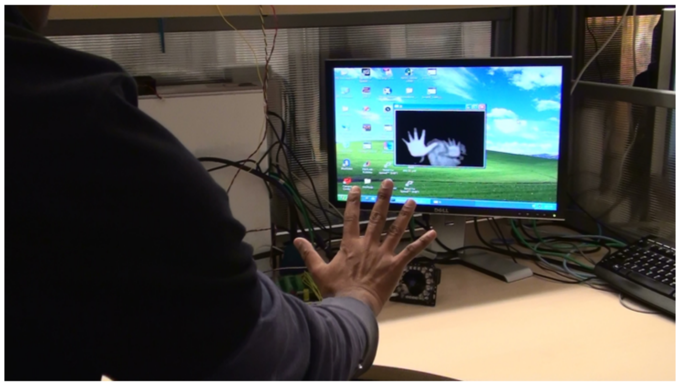
Fig. 2. Grayscale Depth Data Obtained From 3D Camera.

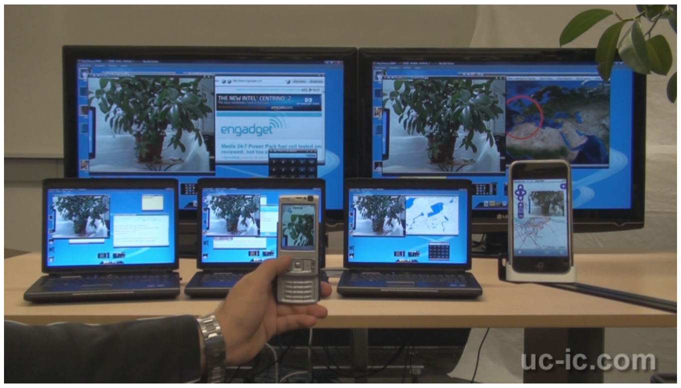

Fig. 3. UC-IC Photo Viewer Collaborative Session Between Seven Users.

to use from their home, and the distributed nature of our collaborative environment ensures that the system can scale as demand grows.

III. Architecture

The architecture of the proposed system consists of a 3D camera that sends depth information to a PC containing a camera driver and a web browser. The camera driver processes the depth data and is accessed by the application within the web browser to determine the latest gesture states. The web browser displays the collaborative UC-IC gesture-driven environment, which is typically hosted on a remote web server and contain application content distributed among several servers.

A. *Gesture Processing Architecture*

The gesture-processing technology used in our approach consists of a new high-resolution 3D camera for real-time depth-per-pixel measurements based on variable gain and gating techniques. The depth-sensing camera uses a combination of Phase measurements and Time-of-Flight (TOF) measurements to extract a depth image from the environment. It works by projecting a pulsed near-infrared cone of light onto the scene and shuttering the returning light in order to generate 256-level grayscale images within which the gray level is proportional to the depth of the object points (brighter pixel values indicate an object that is closer to the camera).

The 3D camera is an embedded system made up of an electronic board for the image sensor, a controller board, an illuminator board, and an image processing board. The controller and the image processing are synchronized to produce the 8-bit depth map from the image sensor output. The camera produces 3D images at the speed of 30 frames per second, with almost no delay in regards to the moves captured. The functioning 3D camera can be seen in Figure 2.

The custom depth-sensing technology is discussed in greater detail in [12].

B. *Collaborative Gesture Environment Architecture*

The UC-IC Collaborative Web Client Platform is a web-based implementation of the desktop metaphor, consisting of familiar windows, icons and menus. The entire environment is built on top of a collaborative platform that allows any window to be "sent" to another user. For the user, this is done simply by dragging a window onto an icon on the web-based desktop representing the user who is to receive the application (and who must acknowledge a dialog to accept it). This sending process is unique in that the entire application logic, in addition to the current data within that application, is sent as part of the window. Sending a map application, for example, means that the receiving user not only receives the application logic (the user interface and geographical functions called by the buttons), but also the current data and state of the application (the exact position where the map is centered). This is made possible by an advanced XML-based syntax and dynamic resource loading techniques (AJAX-Push) that have been described in other works [13].

It is important to note that this concept of "sending" does not necessarily mean that the user doing the sending no longer retains the application. Rather, by draggin the user icon to the application window, both users can have the same window open as part of a collaborative session. The inherent collaboration built into UC-IC ensures that any actions performed within that application are automatically synchronized to the other user. For example, any panning or zooming of a map will be communicated instantly to the other user's browser so that, as much as possible, both users always see the same application state. The environment was named "UC-IC" to highlight these collaborative characteristics. Figure 3 shows seven different users taking part in a UC-IC collaborative session for the photo viewing application.

Real-time collaboration on UC-IC is supported for applications programmed in, or able to communicate through, DHTML (AJAX). This makes it easy to develop new collaborative applications for the environment since this includes a variety of web-based technologies, including JavaScript, Adobe Flash and Java. Existing applications built on the UC-IC platform include a map application, a videoconferencing application, a collaborative video player, a synchronized photo viewer, and a drawing application that can be reused to add annotations on top of other UC-IC applications.

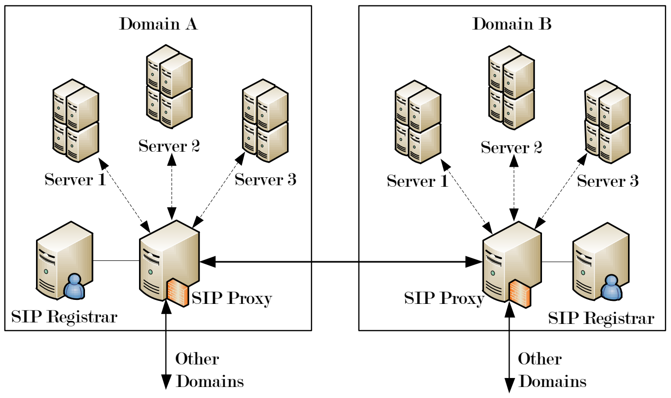

Fig. 4. Two UC-IC Domains Communicating Through SIP.

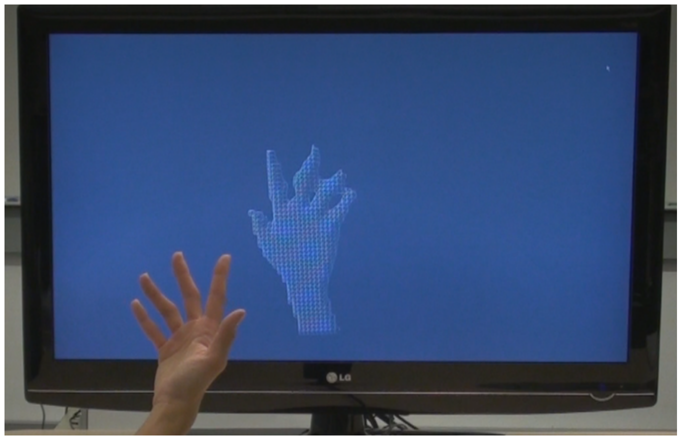

Fig. 5. Real-Time Depth Data Represented as a 3D Mesh.

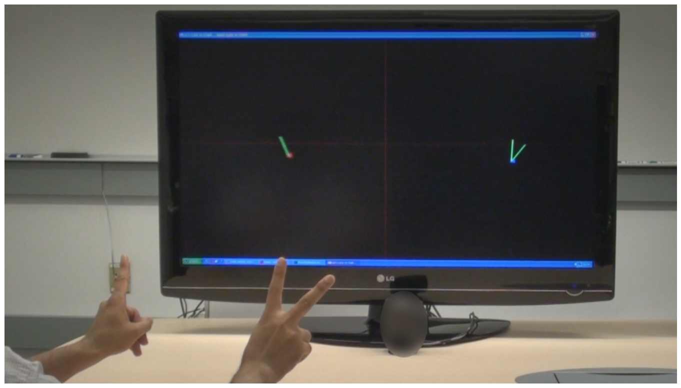

Fig. 6. Detecting the Number of Hands, Fingers and Finger Angles.

The UC-IC application server includes the ability to bridge individual servers into a cluster of servers in one domain, as well as to bridge servers from different domains. This is achieved by using the Session Initiation Protocol (SIP) as the connection protocol between the different servers, allowing users from different domains to be invited to - and thus participate in - the same collaboration session.

Figure 4 shows an example with two clusters of servers, each representing a different domain consisting of three servers. The domains are connected together through the SIP server of each of the clusters. Each distinct server may contain different applications and data that will instantly become available to other users invited to a collaborative session or who were sent an application window.

IV. Implementation

We now discuss the implementation of the novel gesture control system that uses the 3D camera and web-based environment introduced in the previous section.

A. Gesture Processing Implementation

The grayscale data obtained from the depth camera was plotted as a 3D mesh by using standard OpenGL libraries. We adjusted the illumination timings of the camera to obtain a clean image of the hands of the person standing about one meter away from the camera. The detection range of the camera is configurable based on the nature of the application or device controlled, being able to span distance from a few tens of centimeters to a few meters. We applied a threshold to eliminate the data beyond a certain depth (the background). The resulting 3D image of a user's hand can be seen in Figure 5. The user's hands and fingers are captured by the 3D camera and a 3D image is formed.

In order to convert the depth data returned by the 3D camera into gestures useable within a collaborative desktop environment, a component was tasked with applying computer vision algorithms for determining hand gestures to the 3D data. The algorithms detect the convex hull of the hand and the irregularities present when fingers are apart to decide if one, two, or more fingers are shown. The angles of the fingers are also calculated. Figure 6 shows how a test application is able to draw the number and orientation of a user's fingers on both of their hands in real-time.

When the user holds out their hand to perform a gesture, the closest object to camera is the user's hand. Therefore, by tracking the brightest point in the depth data, we can create a mouse cursor. The resolution of the 3D data from the camera can be mapped to the resolution of the screen, with low-pass filtering applied to reduce the jitter introduced by such scaling. A quick movement of the cursor from the right to the left can then be detected as a flicking gesture. We can detect a second hand in a similar fashion and map it to a second cursor. We therefore end up with a large variety of possible hand gestures that we can use to interact with computer applications and electronic devices. Based on these gestures, we can define a natural language that users find to be the most convenient for interacting with a collaborative web-based environment.

B. Collaborative Gesture Environment Implementation

As was mentioned in Section III, UC-IC is a web-based application platform which provides a desktop environment as well as components for application development and management. The DHTML-based user interface provides wrappers for delegation to and from other browser-contained technologies such as Adobe Flash and Java Applets. Thus, AJAX

(JavaScript) components provide the collaboration layer used by applications regardless of the programming language used to build it.

Java Applets are of particular interest as they have the ability to communicate with the driver of the 3D camera connected to the user's PC. We therefore created a UC-IC application that uses the Java Native Interface (JNI) programming framework to access the processed gesture data from the camera driver. This includes details about the current frame such as the number of hands present and their position, number of fingers on each hand and their angles, and if a flicking gesture is taking place.

Since UC-IC is a custom desktop-like environment, two new cursors that mimic the behavior of a regular mouse were introduced and assigned to follow each user's hand. Introducing two functional mouse cursors in the Microsoft Windows environment is much more difficult, as the operating system and all its applications were generally designed to be operated by one mouse cursor. In UC-IC, however, we can draw two objects that represent cursors and have access to the entire virtual desktop environment within the user's browser. Through its access to the DHTML layer, the Java Applet therefore executes JavaScript code to update the positions of each cursor instance and to signal events such as if a click is taking place. We have defined the presence of no fingers to indicate a *clicked* state, while the presence of an upward pointing index finger allows for regular cursor movement. The environment can then respond accordingly to a hidden finger, such as by allowing the user to drag a UC-IC application window if the cursor is on the title area of the UC-IC window.

If a flicking gesture has been detected, a *next* or *previous* command is sent to the UC-IC environment depending on the direction of the flick. This command affects any active slideshow or photo viewing application that the user may have opened. By using the angular finger data, a *thumbs down* gesture was assigned to call the *close window* operation.

We have modeled our natural gesture language to use gestures users are familiar with from *multi-touch* environments such as that of the iPhone. The *pinch-to-zoom* gesture that used are already accustomed to can easily be translated to our touch-free environment by detecting if both hands indicate a *clicked* state and are moving in an inward or outward motion while positioned over the photo viewer application of the UC-IC environment. It is important to add support for gestures such as these, as users may instinctively and naturally assume that the operations exist when they see the ability to control two cursors at the same time.

Once the user is able to manipulate the UC-IC environment by using just their hands, the collaborative nature of each UC-IC workspace means that users can instantly be connected to all of their online friends and co-workers, with which they can share entire applications and the data contained within those applications in real-time. This means that a flick of the wrist not only changes the photo in the photo viewing application within the user's own browser, but also affects the photo viewing applications of any other users that were invited to the collaborative session for that application. The Java Platform, Enterprise Edition (JEE) server-side and AJAX client-side implementation details that make such web-based synchronization possible across distributed servers have been discussed further in [13].

V. Results

A. Gesture Processing Results

The ability to reliably determine if a user is showing one or more fingers has not been possible with non-depth-based camera technology. Eliminating the background data from the depth data returned by the camera offers several advantages over existing solutions, such as those mentioned in Section II. For example, users do not need to perform gestures against a white or stable background for gestures to be detected correctly. In fact, people can be moving around behind the user without affecting the accuracy of the gesture processing.

The amount of data processed by the gesture detection algorithms is also minimized since the algorithms only need to run on a small subset of the data returned by the camera. This allows the system to run in real-time with very little computational burden on the user's PC and with practically no lag or delay between the user's actions and the on-screen response. Additionally, the "gorilla arm" effect is diminished as users can perform gestures quickly.

Since the implementation reliably returns if the user is showing one or more fingers, gestures that were previously not possible to be determined in a robust way can finally be used as part of a system targeted at the typical home user for interacting with each other through a gesture-based collaborative environment.

We have therefore found that depth-based cameras are optimal for hand gesture processing. The recent drop in the price of 3D cameras to the level where they are comparable with high-end webcams indicates to us that there will be an increasing number of people using gesture control in the future. The convenience of having to position just one camera instead of needing a sophisticated laboratory setup is also ideal for consumers and for use in their homes. By using the depth data from a 3D camera, gesture detection algorithms are far less error prone and do not need to be calibrated to a specific user. It also offers flexibility when defining the gesture language for users. The far improved reliability of such processing means that gestures can be easier to learn, easier to remember, less awkward to use, and generally more immersive and natural for the user. There are no special gloves or markers for the user to wear; users can simply walk up to the camera and instantly take control of a mouse cursor by holding up their hand.

Figure 7 shows a Java-based application running on Microsoft Windows that correctly identifies the gestures the user is performing and gives the user access to the Windows mouse cursor. The user is able to play Solitaire by moving their hand to move the on-screen cursor. Hiding their index finger performs a click operation, which can be used to drag the cards to the necessary positions. By allowing such gestures to

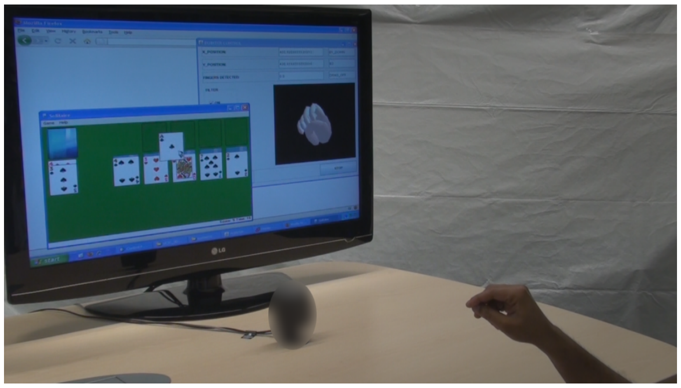
Fig. 7. Controlling the Mouse Cursor in Microsoft Windows.

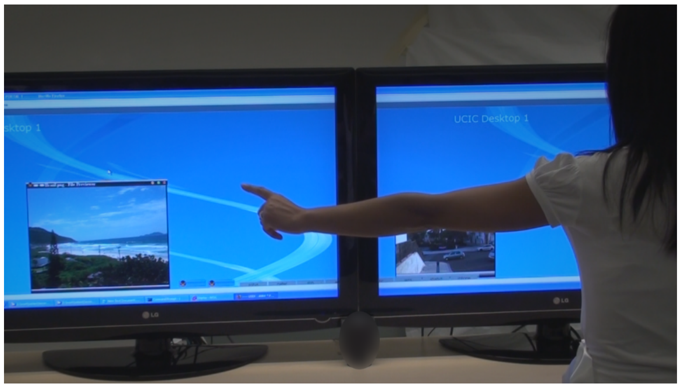

Fig. 8. Dragging an Application Window Across Multiple Monitors.

affect a web browser, we can give the user multiple mouse cursors and a window-based environment containing multiple collaborative applications.

B. Resulting Collaborative Gesture Environment

To evaluate the system, we asked eight friends and colleagues to perform a usability study. The users were tested in pairs so that they can fully evaluate the collaborative features of the environment (although the UC-IC environment supports far more than two users collaborating on the same application). They were given three tasks that cover several major features of UC-IC.

Tasks were tested for about five minutes with each user standing in front of one of two different 3D camera setups. One test setup consisted of two large high-definition televisions connected to the same computer and configured to display a spanned desktop. Maximizing the web browser containing the UC-IC environment therefore created a very wide online desktop that could contain many application windows. The 3D depth camera was connected to the PC and set up between the monitors to pick up the user's hand gestures. The Java Applet was started within the UC-IC environment to enable the gesture-based control. The second test set-up was placed at another PC several meters away so that users could send windows to each other and collaborate on applications. Each PC was running a separate instance of the UC-IC application server to ensure that the SIP-based session establishment and web-based communication functions correctly across distributed domains.

The first task consisted of opening a picture slideshow viewing application by clicking on its icon, flicking next/previous a few times through the images, and then sending that application (containing the image) to another user by dragging the application over the other user's desktop icon. Users were surprised by how quickly the on-screen cursor responded to their moving hand and that they were able to navigate menus and open applications with relative ease. One user needed to adjust the camera to point it slightly lower in order for their clicking gesture (hiding of the index finger) to be detected consistently. Users found the flicking gesture to be a very casual and relaxing way of navigating photos. Although it often took users more than one try to drag an application to another user's icon, they quickly adjusted to the environment and were able to send photos back and forth with little difficulty.

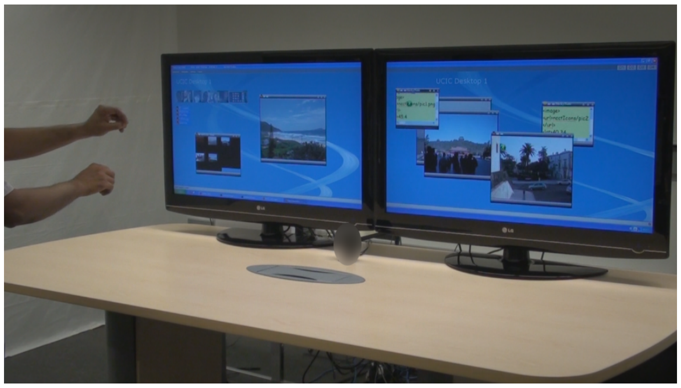

Fig. 9. Controlling Two Cursors to Move Two Windows at the Same Time.

For the second task, we asked users to start a collaborative session with the map application (this time by dragging the user's icon to the application window) and to pan around the map before finally closing the window by performing a thumbs-down gesture. Users were able to invite each other to collaborative sessions without many problems since they were adjusted to the feel of the cursor after having completed the first task. Users were able to pan the map by dragging it with their hand and the others in UC-IC would instantly update to reflect the new location.

The third and final tasks required users to use both hands to move two windows at the same time (as shown in Figure 9) and to resize a photo viewer window. Users exclaimed that, by using two hands, they felt more productive and that the two-handed resizing gesture felt natural and familiar. One difficulty users encountered was that crossing their arms could cause the left cursor to jump to the right hand, or vice versa. This problem can be resolved with further work on the tracking aspect of the gesture processing software.

Some of the tests were filmed and can be seen in the second half of the YouTube video at [14]. Additional experiment shown in the video include the ability to drag applications

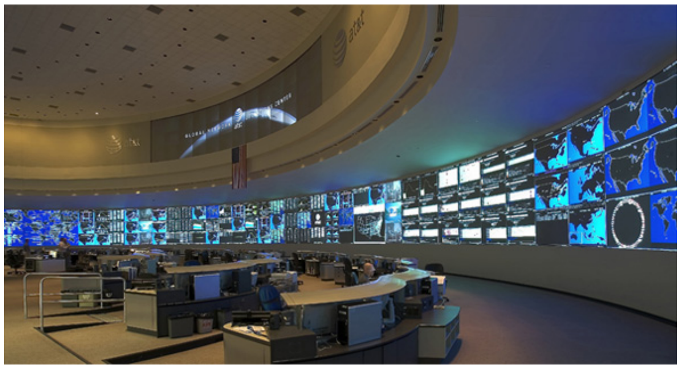

Fig. 10. Display Wall of the AT&T Network Operations Center.

to users on mobile devices (since UC-IC is a browser-based environment and can therefore be accessed from mobile web browsers) and the ability to drag applications containing data onto other applications to create *mashups*, such as a map containing photographs.

By performing these tasks, users were completing useful and compelling scenarios that are already part of many users' lives (such as sending a photo or map to someone). Users are able to move information and manipulate data in a much more natural way by using gestures, and users expressed a feeling of empowerment and productivity that they felt while collaboratively sending the applications with gestures.

C. Reflecting on the Results

While performing the user tests, we observed that moving windows by using gestures instead of the traditional mouse is much easier when large, high resolution displays are involved. Increasingly, users are using large displays, which are becoming slow and akward to navigate with a regular mouse. More and more companies are also setting up control center environments containing hundreds of individual displays, such as the AT&T Network Operations Center in Figure 11. Managing the content on these displays by using the common Barco Apollo [15] software is a tedious process, requiring operators to constantly switch between multiple workstations in order to display the necessary information. By using gestures and the UC-IC collaborative environment, multiple users can be controlling the windows that appear on the large video wall and the windows can be moved around the room by simply pointing to the desired locations. By taking advantage of the distributed nature of UC-IC, such setups can even integrate disparate applications and data from multiple headquarters and teams that a company may have.

We can also see such a system being used in an operating room scenario. Surgeons who are typically unable to touch any unsanitized surfaces can now access and navigate important data within UC-IC by using hand gestures. UC-IC can allow for the seamless display and control of multiple medical systems and present a heart surgeon with interactive information such as the live x-ray location of the pressure wire, the actual pressure measurements, the patient monitor and an angiographic roadmap. This information is typically displayed on separate monitors. Figure 11 shows how such a system may function.

VI. Conclusion

Gesture-based technology has gained significant academic and commercial interest lately with the goal of allowing users to use hand gestures to control computer interfaces in a productive and collaborative way. We have shown how the depth data produced by a custom 3D camera can be used to successfully track hands, fingers and flicking gestures within a distributed and collaborative web-based environment called UC-IC. Existing collaboration products do not connect both users and applications from different domains. Until now, it has not been possible for a user from domain A to be able to collaborate on applications being used by users in domain B where both users are running the full application code and have access to the actual application data.

Our touch-free multi-cursor gesture control solution is realistic for home use since it does not require a specialized lab setup, reliably detects and tracks easy-to-remember gestures, and can be used to complete useful tasks such as sharing photos within a collaborative online environment. Our test users were excited about the benefits offered by gesture controls and how they can be used to complement (or even replace) the traditional keyboard and mouse in many scenarios.

We plan to explore many more potential uses of this technology in the future, including the processing of full-body gestures. In addition, Windows 7 recently added "multi-touch" related features which we would like to make touch-free. This is a strong indicator of the gesture-oriented direction in which human computer interfaces are heading.

References

[1] B. A. Myers, "A Brief History of Human Computer Interaction Technology," in *ACM Interactions*, vol. 5, no. 2. New York, NY, USA: ACM, March 1998, pp. 44–54.

[2] W. T. Freeman and C. D. Weissman, "Television Control by Hand Gestures," in *Proc. of Int. Workshop on Automatic Face and Gesture Recognition*. IEEE Computer Society, 1995, pp. 179–183.

[3] Q. Cai and J. Aggarwal, "Tracking Human Motion Using Multiple Cameras," in *Proc. of 13th Int. Conf. on Pattern Recognition*, vol. 3. IEEE Computer Society, August 1996, pp. 68–72.

[4] Z. Jun, Z. Fangwen, W. Jiaqi, Y. Zhengpeng, and C. Jinbo, "3D Hand Gesture Analysis Based on Multi-Criterion in Multi-Camera System," in *ICAL2008: IEEE Int. Conf. on Automation and Logistics*. IEEE Computer Society, September 2008, pp. 2342–2346.

[5] A. Utsumi, T. Miyasato, and F. Kishino, "Multi-Camera Hand Pose Recognition System Using Skeleton Image," in *RO-MAN'95:Proc. of 4th IEEE Int. Workshop on Robot and Human Communication*. IEEE Computer Society, July 1995, pp. 219–224.

[6] P. Mistry and P. Maes, "SixthSense: A Wearable Gestural Interface," in *ACM SIGGRAPH ASIA 2009 Sketches*. New York, NY, USA: ACM, 2009.

[7] A. Wilson and N. Oliver, "GWindows: Robust Stereo Vision for Gesture-Based Control of Windows," in *ICMI '03: Proc. of 5th Int. Conf. on Multimodal interfaces*. New York, NY, USA: ACM, 2003, pp. 211–218.

[8] M. R. Thissen, J. M. Page, M. C. Bharathi, and T. L. Austin, "Communication Tools for Distributed Software Development Teams," in *SIGMIS-CPR '07: Proc. of ACM SIGMIS CPR Conf. on Computer Personnel Research*. New York, NY, USA: ACM, 2007, pp. 28–35.

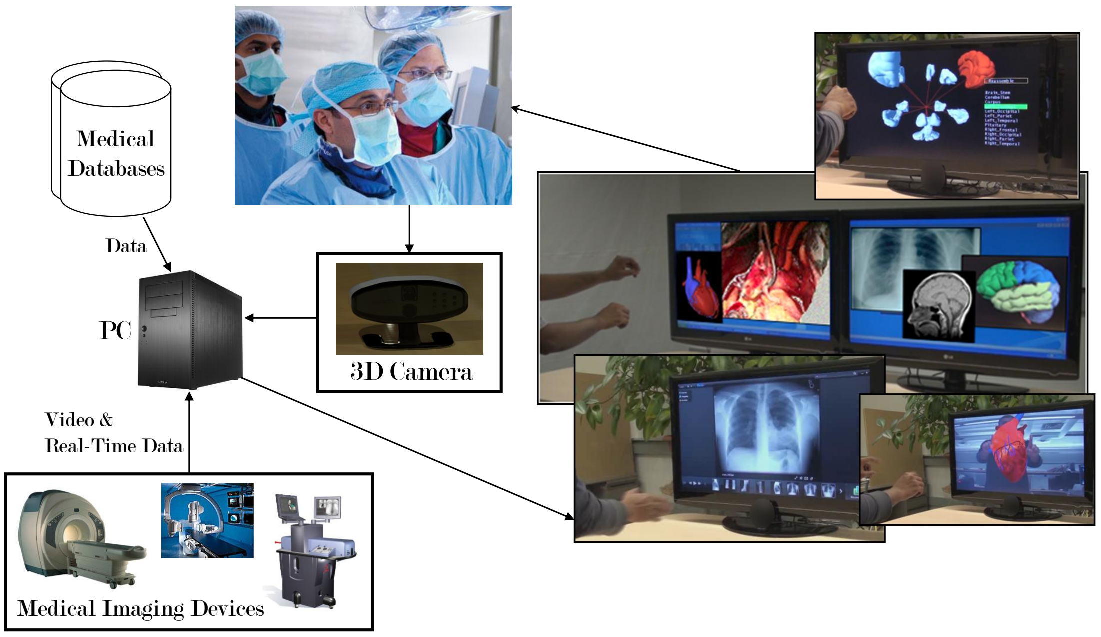

Fig. 11. Architecture of a Gesture Controlled System for the Operating Room.

[9] W. Wang, "Powermeeting: GWT-Based Synchronous Groupware," in *HT '08: Proc. of 19th ACM Conf. on Hypertext and Hypermedia*. New York, NY, USA: ACM, 2008, pp. 251–252.

[10] (2010) Xbox.com Kinect. Microsoft Corp. [Accessed: September 2010]. [Online]. Available: http://www.xbox.com/en-US/kinect

[11] M. Moyle and A. Cockburn, "Gesture Navigation: An Alternative 'Back' for the Future," in *CHI '02 Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2002, pp. 822–823.

[12] D. Ionescu, B. Ionescu, S. Islam, and C. Gadea, "A New Method for 3D Object Reconstruction in Real-Time," in *ICCC-CONTI: 2010 Int. Joint Conf. on Computational Cybernetics and Technical Informatics*, May 2010, pp. 649–654.

[13] R. Dagher, C. Gadea, B. Ionescu, D. Ionescu, and R. Tropper, "UC-IC: A Cloud Based and Real-Time Collaboration Platform Using Many-to-Many-on-Many Relationship," in *Proc. 8th IEEE I2TS*, May 2009, pp. 9–17.

[14] (2008, November) Look ma, just hands! Mgestyk Gesture-Based Gaming and more. Mgestyk Technologies Inc. [Accessed: September 2010]. [Online]. Available: http://www.youtube.com/watch?v=FZyErkPjOR8

[15] (2010, September) Apollo Video Wall Management Software for Windows. Barco Inc. [Accessed: September 2010]. [Online]. Available: http://www.barco.com/en/product/631