Eye-movements and Voice as Interface Modalities to Computer Systems

Mohsen M. Farid* and Fionn Murtagh
School of Computer Science, Queen's University Belfast.

ABSTRACT

We investigate the visual and vocal modalities of interaction with computer systems. We focus our attention on the integration of visual and vocal interface as possible replacement and/or additional modalities to enhance human-computer interaction. We present a new framework for employing eye gaze as a modality of interface. While voice commands, as means of interaction with computers, have been around for a number of years, integration of both the vocal interface and the visual interface, in terms of detecting user's eye movements through an eye-tracking device, is novel and promises to open the horizons for new applications where a hand-mouse interface provides little or no apparent support to the task to be accomplished. We present an array of applications to illustrate the new framework and eye-voice integration.

Keywords: Eye Tracking, Visual User Interface, Visual Navigation, Vocal User Interface, Voice Commands, Graphical User Interface, ASL-504, Non-Haptic Devices.

1. INTRODUCTION

The 1968 science fiction epic and classic film "2001: A Space Odyssey" was considered a source of motivation for speech and speech control research, both on the academic and industrial fronts. Hollywood has also gone as far producing movies where a computer tracking eyes could tell what a person is seeing. It was an artist's dream leading to substantial research and funding to understand human vision and voice as modalities of control of a computer system. Computer systems interact with the outside world in the form of specialized input/output subsystems which perform basic processing in a limited communication band. Humans on the other hand have much wider bandwidth for communication not only with one another but most notably with machines. A communication channel between a computer and a human is not isotropic [1]. Unlike humans, computers have a constrained interface through which they can interact with the outside world.

Human-computer interaction addresses [10, 11] five primary interaction modes: menu selection, form fill-in, command line, natural language and direct manipulation. In a typical user environment, it is customary to find one or more of such modes. Direct manipulation, however, by far has distinguished itself from the rest as it enables users to easily and naturally select and manipulate an object in isolation or in conjunction with some other object with the least amount of effort. In a graphical user interface environment, direct manipulation is mostly accomplished via the use of a pointing device such as the electromechanical mouse.

Direct manipulation requires user interface designers to rethink their strategies in creating visual representations that are predicable and controllable and including interface modalities that otherwise might not be available. The success of the use of pointing devices such as the mouse was in its ability to create the parallelism with human skills such as pointing, grabbing and moving objects. But as we perform such tasks we employ the visual elements that actually create the mental image of things that we need to do to accomplish such tasks. In other words, our hands do follow out eyes.

Eye gaze, within the direct manipulation paradigm, could easily afford possibilities of interface with computers as yet another input device, mimicking a pointing device, such as the electro-mechanical mouse. Eye gaze draws its strength from manipulating non-textual objects, in an environment where eye fixations are received as a continuous stream of input data and which are interpreted on a real time basis. Unlike pointing devices where events take place on the basis

* Corresponding author: m.farid@qub.ac.uk; phone +44 28 90 27 46 21; School of Computer Science, Queen's University Belfast, Northern Ireland, BT7 1NN, United Kingdom.

of events, such as pressing a button, eye gaze tracking depends on receiving a continuous stream of data that needs to be manipulated, processed and interpreted, as processing of other tasks continues. The programming environment that serves such a paradigm is multi-threaded.

Eye gaze trace data have relatively simple structure; the x and y coordinate of, and fixation on, the screen, and time information. But to make any functional sense of data, the data has to be processed on a spatio-temporal basis to allow for modeling of events to fit that of a pointing device.

2. EYE GAZE TRACKING

Eye tracking technology is, to a great extent, limited to laboratory environments because of the high investment associated with its acquisition. Eye tracking equipment currently has a voluminous control box which provides all hardware and firmware support that the camera and the host computer require. But to use such equipment in control, the setup needs an additional computer to process the data. The eye tracking setup also requires careful calibration with every new subject. The technology has not yet arrived at the stage where it can serve in a production environment.

Despite its shortcomings, it has been used successfully in many applications. It has potential applications such as in operating room activities, with surgery planning, execution, monitoring, etc. that would materialize as the technology matures. Additionally, eye tracking [6, 7] has been used to address the needs of motor handicapped persons and to increase their productivity and their empowerment. Furthermore, target tracking and acquisition, and reduction of fatigue and potential injuries have been discussed in the literature [8, 9] as other potential uses of eye gaze tracking.

Farid et al. [13] and Murtagh [14] have devised, using C++ and MFC, a methodology whereby simple single click events can be used to evoke certain contextually based actions such as controlling multiple video streams in a web browser. The same techniques were used to decompress, at varied compression rates, large medical (orthopaedic) images from 8.4 MB to 568 KB.

Sibert et al. [2, 3] have made side by side comparisons between the performance of an eye tracking device and the mouse under the same experimental settings. They have determined that object selection is faster using the eye tracker than the mouse. They have also created an empirical metric to measure the performance of on screen selection of an object. Furthermore, if eye gaze tracking does not show observable delay in object selection and "breaks even" with the performance of a mouse under the same conditions, then eye tracking performs acceptably well. We have calculated the velocity components along the x and y direction and found the maximum values to be around 19.0×10^3 and 17.5×10^3 pixel per second for v_x and v_y respectively, which is considerably larger than the speed components of the electro-mechanical mouse. We additionally find visual target lock-ins at higher amplitudes and higher speeds to be more accurate than those using hand-controlled mice, which agrees with Sibert et al. Furthermore, we, verifiably, find that target over- and under-shoots are much smaller that their counterparts using mice at higher speeds.

The aforementioned discussion does not make any general case in favor of visually controlled events over those using the conventional mechanical mouse. In fact the visual modality of interface cannot fully replace the mouse as an input device despite its superior performance in certain aspects of its behavior. Therefore we see that the visual modality of interface could be employed as complementary to the electro-mechanical.

It has been documented by a number of researchers such as Lankford [6, 7] and Zhai et al. [8, 9] that the use of eye tracking technology by the disabled provides them with the ability to communicate with the environment that surrounds them. While this and other applications are examples of valid use of the technology, we think that eye gaze tracking technology could enhance both the productivity and efficiency of a surgeon in an operating room. As such, one of the requirements that we had to impose on the architecture of our eye tracking systems is that the eye should be allowed to manipulate screen objects without resorting to visual which in effect will distract the subject's attention.

Lankford [6, 7] has used assistive visual decision methods to complete visual tasks. Lankford, using the ERICA system, has demonstrated that additional visually evoked mouse events can further be generated and coupled with visual iconic cues to represent mouse functionality such as dragging and double clicking, etc. The most basic operation in ERICA is a "visual" click, where once captured, the subject is then, presented with a set of on-the-screen, iconic menu choices showing other images of different mouse functions such as "Left Double Click", "Left Drag", etc.

Zhai et al. [8, 9] alternatively have asserted that eye gaze has fundamental limitations and that it is unnatural to overload the visual perceptual channel (eye) with motor control tasks. They further proposed the "Manual And Gaze Input Cascaded (MAGIC) pointing". In MAGIC, "ballistic" cursor movements were left to eye control, where the cursor is moved from one screen location to a region of tolerance in which the final target point is located. Subsequent compensation for over- or under-shoots, that attempts to lock into the final target, could manually be controlled.

Similarly, Sibert and Jacob [2, 3, 4, 5] have divided the subject's monitor into two areas: the first is used for object selection using the eye tracker and the other area displays information about the selected object.

2.1. Mouse Emulation: Introductory Remarks

In attempting to address functionality of "eye-mouse", researchers have tried to mimic all or most of the range of mouse functionality available in Microsoft Windows and maybe other operating systems. Obviously, graphical user interfaces within an operating system environment define a range of mouse functions and leave it to applications software to override and/or overload some of these functions, i.e. to add new features to these applications.

For example, tapping the left mouse button and holding it while moving (or dragging) over text in Microsoft Word results in marking the text for another operation that needs to follow, e.g. deletion. Performing the same operation in the desktop environment of MS Windows results in "dragging and dropping" the chosen object(s), i.e. moving the object(s) into another folder.

Electromechanical mice or their optical counterparts are convenient, accessible and, more importantly, are fine tuned to address information at the pixel level. The eyes, as components of the ocular-motor system, do not have the same range of capabilities, and as such cannot handle *all* the motor functions imposed by the implementation of the electromechanical mouse, without being inefficient and/or strenuous. While a mouse can be positioned on a pixel, the eye is unable to fixate on a point voluntarily. The eye is constantly moving about the object of interest. The eye's constant move is termed saccadic motion. Saccadic motion is rapid 2-dimensional Brownian-like motion. While we can see a pixel, it is not possible to fixate on that pixel, in the same way as we can "hold" it using a mouse

Implementations of "eye- mouse" capabilities are based on capturing eye fixations within the confines of monitor boundaries. To implement a comprehensive set of mouse functionalities, a designer had to resort to one or two general approaches. The first, [6, 7] is to capture a visual mouse click and then display iconic menu entries which the user could select from to complete the task at hand. In the second approach [8], the subject completes the task by supplementing the system with a mouse or keyboard tap(s). In either case, the visual continuity of the task is interrupted by starting another task whose purpose is to control the task that the subject has originally started. Once completed, the user may return to the application domain. These series of steps may be repeated as many times as deemed necessary. The cost of the first approach is strained eyes and possibly frustration; and in the second the system requires manual handling of events which may not justify the whole effort of using eye tracking for control.

2.2. Dissection of the Basic Mouse Events

Raskin [15] defines the tap of a pointing device (or the Graphic Input Device - GID) as "the act of pressing and releasing a key or a switch without any intervening action". This definition applies to either a keyboard or a mouse. A *click***,** however, consists of a GID tap and a record of the location of where the tap took place. A *double click***,** consequently, is to click twice in a quick succession without "changing" the location of the second click. We note that the time between the two taps is user configurable but within some system limitations. We also note that the environment allows for some tolerance, i.e. a small displacement between the two taps, which can be ignored. A *drag-and-drop***,** then, is a press down of a mouse button, followed by a move to another location, followed by a release of the button of the same pointing device.

We consider one or two button pointing devices (3 button pointing devices are only considered when the middle button is not used or disabled). A mouse button may be in one or more of the following states: Down (D), Up (U) or Move (M). The Move state is associated with a change of position p0 to position pl. One could list the mouse states for each button as in the following table 1.

Event/State	Representation	Comments
Tap	D-U	Ignore position
Click	D-U-p0	Cannot ignore position
Double Click	D-U-p0 + D-U-p0	The time differential is normally user configured, position is ignored
Drag-and-Drop	D-p0-M-U-p1	Displacement in position is recorded but time is ignored.

Table 1. Basic Mouse functions.

Applications and operating environments may implement services to associate features with some or all of these events. Applications may also associate these four states with keyboard keys such as shift, ctrl or alt in conjunction with other keys to reproduce some of the mouse functionality. We do not consider mouse wheels as they can easily be represented by any combination of the above events.

We wish to make the following observations regarding the eye emulation of the mouse:

- In experiments where the eyes (as a perception channel) are used to fully emulate a pointing device, we observe a considerable compromise of the functionality which renders the effort counter-productive.

- Application programs would find it necessary to override or overload some mouse functionality to make additional features available in an attempt to make the system more accessible to users.

- There are instances where using either the mouse or the keyboard is counter-productive. Surgical procedure, motor handicap, target tracking and avoidance of injury are all examples of these situations. In such applications, mouse and keyboard interaction are either not desirable or may be considered complementary because manual direct manipulation is not necessarily possible. In such cases as surgical procedure, it might have detrimental or life threatening results if the surgeon left the task at hand to attend to the task of controlling what is being displayed on a computer monitor. Additionally, depending on the other support staff in the operating room, it might not be the optimum way of handling such a task and may have other needless drawbacks.

  In the case of the severely handicapped, the ability to handle fine motor control tasks such as using a mouse or even a keyboard might not even be a choice.

  In all these examples the chief problem of using a pointing device or keyboard is the interference with oneself to accomplish the task without resorting to others to have the task complete. Additionally this interference minimizes users' ability to directly manipulate the objects in question. Where applicable and where menu selection does not interfere with that task at hand, limited screen choices may be made available using solely the eye mouse.

- Due to the nature of the eyes and the strain that they may experience in trying to accomplish a large number of tasks for long periods of time, we must try to free up ourselves from overloading the eyes with lots of unnecessary mouse movements.

- It is a sound design practice to provide a basic set of mouse functionality that is, generally speaking, application independent, but yet can be overridden and/or overloaded by the controlling application programs to accommodate the application's own design requirements for the situation at hand.

2.3. Eye Mouse Event Modeling

We use the ASL 504 Pan/Tilt Optics Eye Tracker manufactured by Applied Science Laboratories. We normally calibrate the system at the beginning of each test run to ensure the validity and correctness of the data points collected. We use a 9-point calibration**.** All control experiments are written in Microsoft Visual C++ with MFC, Java and/or Matlab. Data recording takes place via the external port, XDAT serial port using the communication protocol set by ASL, using a 2 monitor workstation running Windows XP. One monitor is viewed by the subject and the other is used by the experimenter. We set the ASL eye tracking frequency to the default value of 60 Hz. We use the "demand" mode where the host workstation requests data from the eye tracker by sending one byte. The eye tracker, in turn, returns an 8 byte message containing the pupil diameter, the x-component and y-component of the point of gaze, all in a 16-bit representation.

To construct a visual mouse click, we define an eye fixation counter that counts the number of collected eye fixations Δn, within a certain amount of time Δt_0 within a certain bounding square of dimensions Δx and Δy. Δn, Δt_0, Δx, Δy are user configurable. To claim a visual click all of the Δn points must be collected within the designated area, $\Delta x \Delta y$. If the subject blinks or moves out of the imaginary square during the designated time interval, the counter is reset to a new start. If the subject successfully completes all Δn fixations within the designated area in the designated time interval, then a visual tap is recorded. To calculate the position of the tap (p_0), we compute the moving average of both the x and y components of all successful fixations.

In our implementation of the eye mouse, the basic assumption we make is the subject in a default state of an "incomplete" drag-and-drop. To complete this operation, the system waits for another click (a visual tap + position i.e. D-U-p1) at another position p_1 ($p_1 \neq p_0$). The second click must occur within a user definable time Δt_1.

When the subject continues to gaze within the same square area of $\Delta x * \Delta y$ for a time period equal to or slightly greater than $2\Delta t_0$ the drag-and-drop state changes to a double click state (D-U-p0 + D-U-p0) and a double click is recorded and a suitable action is carried out .

We have made a design decision to make all parameters Δt_0, Δx, Δy be user configurable to accommodate both the changing user requirements as well as the application interaction environment. We, however, have set default values that are based on usability studies that we have conducted in our laboratory. We have set the Δt_0 to a default of one second in both the Matlab and C++ implementations. In Java the default value of this Δt_0 is set at .75 second. Additionally, the number of points that are needed to define one visual click is user definable but initially set to a default value of 10. These choices seem to provide a comfortable environment for subjects we have used.

All events defined above are reset to their initial values when a blink is encountered or when the eye gets distracted outside the tolerance region $\Delta x \Delta y$. While a click would normally be followed by a drag and drop or by another click to form a double click, there are instances where the user just visually clicks but never follows this click with any additional action. This renders the sequence of events incomplete leading to a reset, where the eye mouse system goes back to its initial state.

In the following section we give examples of how to implement context based applications that use the eye mouse as the primary method of interaction.

3. EYE-MOUSE IMPLEMENTATION EXAMPLES

3.1. Experimental Setup

Target applications for testing our approach address 2-dimensional image manipulation (such as image translation, rotation, compression, decompression, etc.), 3-dimensional object manipulation (e.g translation, rotation, fly-by and fly-through), and web browser based applications to stream audio and video files. The test applications we chose generally depict a sample of applications that may be suitable for eye mouse processing. In all the examples that we show in this paper, all tasks are carried out by the eye with minimal or no motor channel interference. By interference we mean the need to visually navigate through different areas of the screen or to use a pointing device or the keyboard to perform complementary tasks to complete the task at hand.

The ASL 504 eye tracking system consists of a TV/computer monitor, an infra-red sensitive eye tracking camera that tracks the subject's eye, an eye tracking control unit, and two view monitors: one to display the subject's eye as it fixates on the scene and the other to display the image of the scene being viewed by the subject and a cross-hair, superimposed on it, to show the location of eye-fixations as the subject's eye moves as it views the scene. The system also has a host computer equipped with the control and data collection software.

The eye mouse control system adds another PC which hosts all configuration files and control software and is connected to the eye tracking control unit through the 9-pin serial XDAT port. This host computer is also equipped with two video cards, one connected to the subject‘s monitor and the other fused by the experimenter for control purposes

The control panel has buttons for opening and closing and starting and stopping the serial port. It has also a drop down list to select images needed for viewing during an eye mouse session. As fixations are received, the control panel displays the coordinates (x, y, t) of eye fixation on the subject’s monitor in ASL pixels. (ASL has what may be termed “ASL pixels” where the monitor’s view area is divided into a mesh of 260x245 units.)

Of importance to the control panel is the number of fixations Δn_0 needed to effect an eye mouse click. The experimenter has a range values from 1 to 100 clicks within the time interval Δt. The initial image size is also controlled though the size of image scroll bar. The tolerance area defines the size of the rectangle in which all fixations for an eye mouse click are collected. Δx and Δy could be varied independently making rendering the application valuable in a variety of measurement settings.

3.2. Two Dimensional Image Manipulation

To test our eye mouse capabilities we design a simple visual application launcher, Figure 2, which has a hot spot interface that can easily be invoked visually. The hot spots on the visual application launcher (VAL) are divided broadly into 2-dimensional display tools, 3-dimensional display tools and web browser based tools.

Visual Application Launcher		
2-D Image Navigator	Large Image Navigator	Pipe traversal
3-D Manipulation	Compression Decompression	Web Browsing

Figure 1. The Visual Application Launcher

Two dimensional image manipulation tools include image scrolling and translation, image zoom-in and zoom-out. The visual application launcher also includes a hot spot for navigating large images of the order of 16,000x16,000 pixel. Figures 2 and 3 show an example of navigation of a medium size image (3000x3964 pixels) from the European Southern Observatory, ESO [15]. If the size of the image is larger than the allowable window size on the view monitor, image scrolling is activated allowing the subject to visually scroll through the images in all directions. There are four principal directions along which one could move the image. These directions are East, West, North and South. To scroll one should visually click in the direction where scrolling should take place. For example to scroll right to left (i.e. moving west) one should visually click on the right edge of the image or the right edge of the bounding box. Likewise scrolling an image along any of the other principal directions follows the same pattern. We have allowed another set of four operations where scrolling is allowed along the diagonal and off diagonal axes. Diagonal and off-diagonal scrolling is possible by continually visually clicking around any of the four corners. For example to view the north-eastern section of an image, one would visually click on the top right corner of the image or the bounding box.

The same application, in an intuitive way, can zoom in and out of an image. In Figure 2**,** the image has 514x719 pixels.

One would visually click around the center of the image to zoom-in. The image in Figure 4 shows the zoomed-in image whose dimensions are 3000x3964 pixels. Zooming out of an image requires the subject to move his/her eye to the "zoom-out" button and visually click it. Pressing a button external to the image to zoom-out of the image, in this case, does not impose any unnecessary moves and does not interfere with any visual tasks that the subject may be performing at the time.

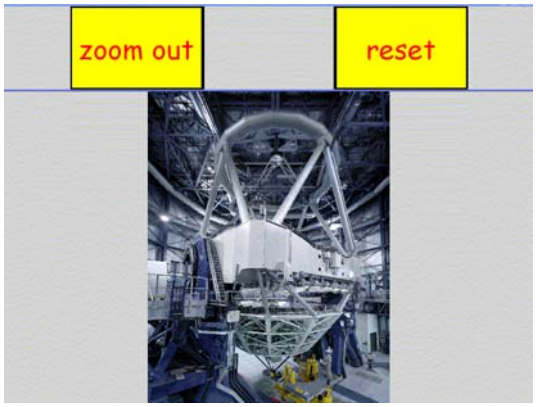

Figure 2. Original image. Image translation and scrolling are visually triggered.

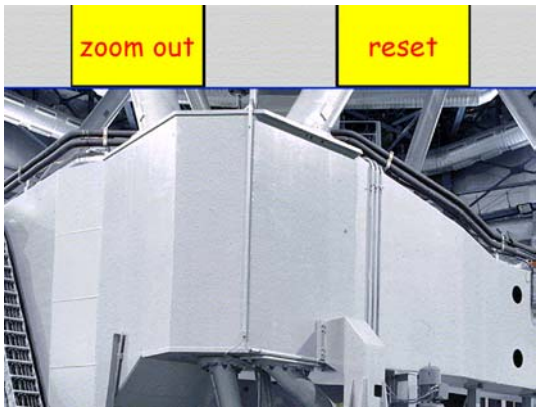

Figure 3. Large image manipulation, with zoom-in and zoom-out functionality.

The application provides a "reset" button whose purpose is to reset the image to the initial conditions at load time.

3.3. Three Dimensional Manipulation

The visual application launcher has the additional capability of allowing the subject to manipulate 3D objects through translation, along any of the three axes, rotation about any of the three principal axes and with fly-by and fly-through, Figure 4.

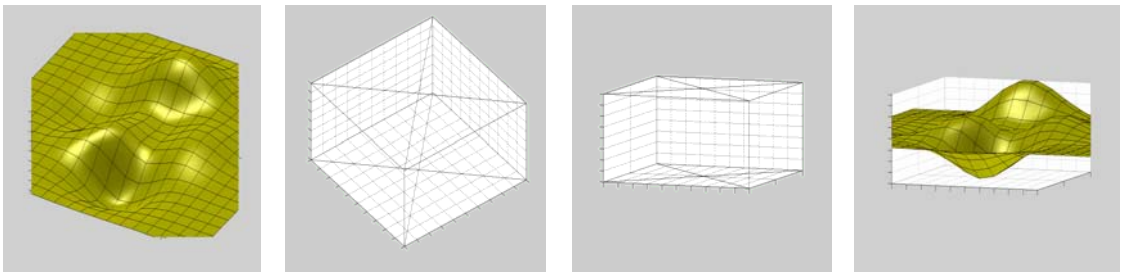

Figure 4; Three dimensional object manipulation

3.4. Web Browsing

Among all the applications that we have addressed in this paper, we find web browsing to be the most challenging of all, primarily because page layout is normally designed by other programmers or content developers who are not familiar with the constraints the eye tracking control imposes. As indicated earlier, eye fixations are not concentrated at a point; rather they follow a Brownian-like motion leading more or less to a spot of significant size. This means that eye fixations cannot be focused on an area any smaller than a certain size. Determining the smallest distance between two close objects, hence determining the smallest spot size, that eye gaze of an individual can select is more or less an art and depends on a number of factors, one of which is the subject's eye. We describe this distance as "eye gaze resolving power". This topic is currently being researched in our laboratory. Farid et al. [13] show examples of a simple web page with easily triggered links using eye gaze. User interest, demonstrated by eye-gaze of sufficient duration on a description panel (a link), causes the streaming video to be activated. If the user continues to watch that panel, the streaming video continues. If the user switches his/her attention to another panel, the previous one is turned off**,** and the panel of interest starts to show video.

3.5. Compression and Decompression

In this application, (see Farid et al. 2002), an image (4096x4096 pixel, 13MB) is digitized at a full size. The image is, then, divided into 16x16 blocks, where each is compressed using a multiresolution wavelet transform. For each resolution level, resulting blocks are then superimposed on the original image. When a block in the low resolution image is visually clicked, the block is extracted and decompressed on the fly.

3.6. Pipe Navigation

Pipe navigator is another application, Figure 5, which enables the subject to visually push a small disk into a 2-D pipe. Navigating the pipe takes place by visually pressing one of four navigation buttons to steer the disk in the proper direction in the pipe. The disk continues to move in the direction of the pressed button. The disk stops moving if it hits a wall. The disk stops when none of the four steering buttons is pressed (also indicating a lack of interest.) In another version of the same application, Figure 6, the disk is steered directly into the pipe without resorting to steering buttons. That version has collision avoidance control which is activated when the disc is within a tolerance zone that is configured as part of the application data.

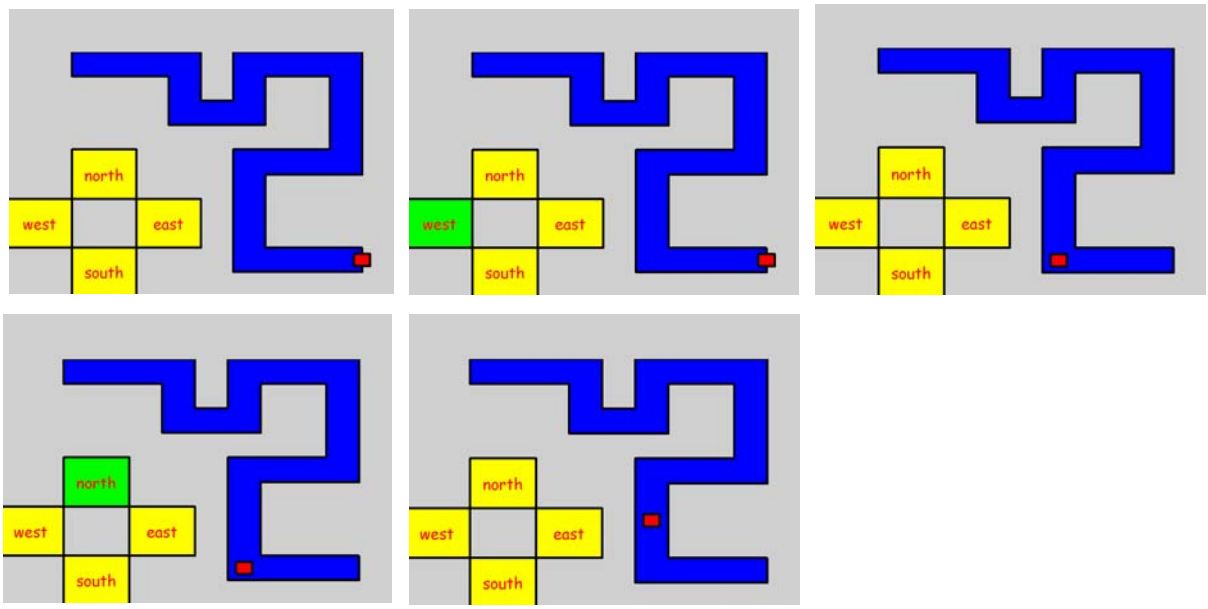

Figure 5. Pipe navigator using steering buttons.

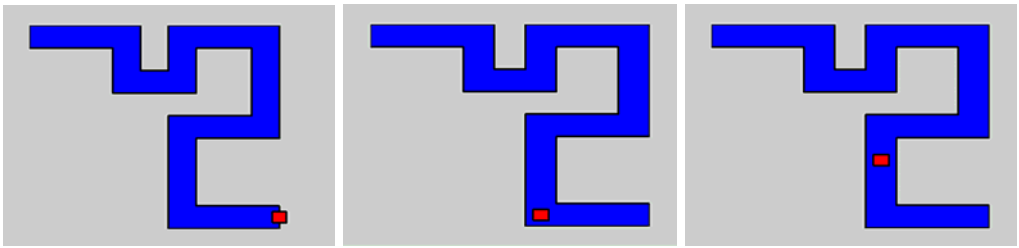

Figure 6. Directly manipulated disk moves into a 2-D pipe with collision avoidance.

4. VOICE INTERFACE

Two way voice communication between a human and a computer, as is the case between two strangers having an unplanned topic of discussion, friends having a conversation about something that they both know and are familiar with or even colleagues discussing a subject matter of confined vocabulary, does not seem to be technologically feasible any time soon.

Human-computer voice communication is still attractive because it is natural for humans to express their ideas in spoken words. Speech is usually presented as sequential events in a well-defined time order, which in itself has an informational content. But this very fact means that the listener has to wait until the speaker completes the thought. We also note that, unlike the visual channel which is localized, speech and audio are not. During speech, utterances are broadcast and all listeners get exactly the same message. Listeners do not have to do anything special to hear the message, it is received anyway. What is done with it falls into the cognitive activities that the listener might want to perform.

Human-computer interface entails a number of activities that all require intensive compute time. Speech synthesis, which can be implemented economically using software or in more dedicated hardware, takes ASCII text, as its input, and produces speech for the output. This is accomplished by using production rules with or without the aid of dictionary lookup. Generated speech can either be synthesized internally or played using pre-recorded human voice. Speech quality, varying in quality, is mostly a function of cost.

In speech recognition systems, spoken words are recognized but not understood. Recognition normally involves pattern matching of the acoustic signal with templates that are stored in memory. Speaker dependence, size of dictionary and signal boundaries all affect the complexity and performance of a speech recognition system. Natural language recognition and generation add to the complexity of such systems.

Stored speech messages and sound are effective in human-human interaction. This is due to the fact that these could be used to enhance communication of ideas among humans with no need to recognize or synthesize speech. Presentation software such as Microsoft PowerPoint uses stored speech and voice for that purpose. Stored speech uses minimum resources and does not impose much in terms of either hardware or software. In fact all modern operating systems have an embedded support for them.

4.1. Use of Voice in the Eye Tracking Domain

In implementing the eye mouse we have avoided using the computer monitor for anything but the display of the most relevant objects that are needed to be directly manipulated. Status and feedback are reserved to stored voice message processing. The user as he/she manipulates objects on the monitor using the eye mouse interface is vocally presented with vocal messages that describe the task at hand. Predetermined voice messages were recorded to replace text messages that would otherwise be displayed.

In the case of two dimensional translation, the user is prompted in the form of the voice back message that the object is "moving east", "moving west", "south east", etc. These messages are stored as simple .wav files. Because of the length of the spoken messages, the files are very small in size and hence do not impose any time constraints to load them and display them.

Subjects, testing this system, have indicated satisfaction with this voice feedback as it does not impose any additional stress in interacting with the system, and did not require more that a basic pair of speakers.

5. CONCLUSION

We describe new approaches in implementing interaction of users with computers equipped with eye tracking systems. The system imposes no strain on the human visual systems as it visually controls the processing of an array of software applications. Additionally, this paper has addressed aspects of the human-computer interface that pertain to feedback by the computer to the human as processing progresses.

ACKNOWLEDGEMENT

The authors wish to thank Mr. Raphael Pezet and Mr. Ian Davis, both of the School of Computer Science, Queen's University Belfast, for help in this project.

REFERENCES

[1] Jacob, R. J. K., Eye tracking in advanced interface design. In Barøeld, W., & Furness, T. (Eds.), Advanced Interface Design and Virtual Environments, pp. 258- 288. Oxford University Press, Oxford, 1995. http://citeseer.nj.nec.com/jacob95eye.html

[2] Sibert, L. E., Templeman, J. N., and Jacob, R. J. K., Evaluation and Analysis of Eye Gaze Interaction. NRL Report, Naval Research Laboratory, Washington, D.C., 2000, http://citeseer.nj.nec.com/sibert00evaluation.html

[3] Sibert, L. E. and Jacob, R. J. K., Evaluation of Eye Gaze Interaction, Proceedings of the CHI 2000, pages 281-288, ACM, New York, 2000. http://citeseer.nj.nec.com/article/sibert00evaluation.html

[4] Jacob, R. J. K., The Use of Eye Movements in Human-Computer Interaction Techniques: What You Look At is What you Get, ACM Transactions on Information Systems, Vol. 9, No3, April 1981, pages 152-159

[5] Jacob, R. J. K., What You Look At Is What You Get: Eye Movement-Based Interaction Techniques, Proceedings of the CHI 1990, pages 11-18, ACM, New York, 1990.

[6] Lankford, C., Gazetracker: Software Designed to Facilitates Eye Movement Analysis, Eye Tracking Research & Applications Symposium 2000, Palm Beach, Florida, USA, ACM, New York, 2000

Research & Applications Symposium 2000, Palm Beach, Florida, USA. ACM, New York, 2000

[7] Lankford, C., Effective Eye-Gaze Input Into Windows, Research & Applications Symposium 2000, Palm Beach, Florida, USA. ACM, New York, 2000

[8] Zhai, S., Morimoto, C. Ihde, S., Manual And Gaze Cascaded (MAGIC) Pointing, Proceedings of the CHI 2000, pages 246-253, ACM, New York, 1999.

[9] Maglio, P., Matlock, T., Campbell, C., Zhai, S., Smith, B.A., Gaze And Speech An Attentive User Interface, Proc. of the Third Int'l Conf. on Multimodal Interfaces, Beijing, China, Oct. 2000. http://citeseer.nj.nec.com/maglio00gaze.html

[10] Shneiderman, B., Taxonomy And Rule Base For The Selection Of Interaction Styles, in Baecker, R., Grudin, J., Buxton, W. A. S., and Greenberg, S., (Eds.), Human-Computer Interaction: Toward the Year 2000, 401-410, Morgan Kaufmann Publishers, San Francisco, 1995.

[11] Shneiderman, B., Direct Manipulation For Comprehensible, Predictable And Controllable User Interfaces, Proceedings of the 2nd International Conference on Intelligent User Interface, Orlando, Florida, USA,1997.

[12] Shneiderman, B. and Maes, P., Direct Manipulation vs. Interface Agents, Interactions Volume 4, Issue 6, Nov./Dec. 1997

[13] Farid, M., Murtagh, F., and Starck, JL., Computer Display Control And Interaction Using Eye-gaze, Journal of the Society for Information Display, 2002, in press.

[14] Murtagh, F., Orthopaedics Example, http://strule.cs.qub.ac.uk/imed.html

[15] European Southern Observatory 1998 - http://www.eso.org/outreach/press-rel/phot- hires.html, and http://www.eso.org/outreach/press-rel/pr-I 998/phot-16-98-hires.jpg.