The VideoMouse: A Camera-Based Multi-Degree-of-Freedom Input Device

Ken Hinckley, Mike Sinclair, Erik Hanson, Richard Szeliski, Matt Conway
Microsoft Research, One Microsoft Way, Redmond, WA 98052
{kenh, sinclair, erikhan, szeliski, mconway}@microsoft.com; Tel: +1-425-703-9065

ABSTRACT
The VideoMouse is a mouse that uses a camera as its input sensor. A real-time vision algorithm determines the six degree-of-freedom mouse posture, consisting of 2D motion, tilt in the forward/back and left/right axes, rotation of the mouse about its vertical axis, and some limited height sensing. Thus, a familiar 2D device can be extended for three-dimensional manipulation, while remaining suitable for standard 2D GUI tasks. We describe techniques for mouse functionality, 3D manipulation, navigating large 2D spaces, and using the camera for lightweight scanning tasks.

Keywords
Input devices, interaction techniques, multi-degree-of-freedom input, rotation, tilt sensing, camera-based input

INTRODUCTION
Many tasks, such as navigation, 3D object manipulation, and image editing [15][6] can require multiple degrees of freedom (DOF) of rotation, zooming, or translation. Conventional mice, however, allow integrated control of only two degrees of freedom at any one time. While 3D or 6DOF input devices can help to provide these missing degrees of freedom, such devices can be ineffective for standard 2D cursor control. Mice can also be augmented with wheels or joysticks for added DOF's [27], but typically these controls are dedicated to secondary low-DOF tasks such as scrolling or panning. All of these devices have limitations in a workflow which may frequently switch between 2D pointing tasks and multi-DOF manipulations. Input devices and interaction techniques that can enhance the directness and degree of manipulation possible in such a workflow thus represent an important area of inquiry.

A promising approach is to evolve the mouse so that it can indeed sense additional degrees of freedom, as exemplified by the Rockin' Mouse [1]. The Rockin'Mouse is a mouse-like tilt-sensing input device which has a curved base with a small flat spot for stability. This simple design, the essence of which we have adopted for the current VideoMouse form-factor, provides simultaneous, integral control [1][13] of 4 DOF while moving the device on a planar surface. This class of *planar multi-DOF devices* has a good balance of properties which are favorable for both 2D and 3D interaction. The flat spot affords constraint of the device to standard 2D translation, yet a subtle tilting motion can activate additional degrees of freedom when needed.

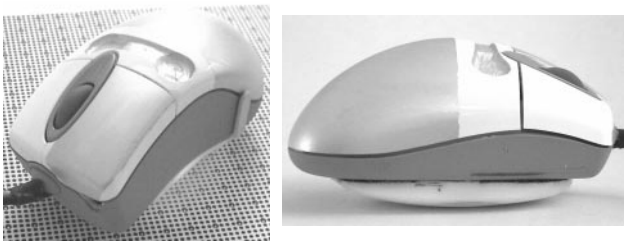

Fig. 1 The VideoMouse uses a camera to sense six degree of freedom motion in a mouse-like form factor. It has a curved base to afford tilting the device [1].

The VideoMouse (*fig. 1*) uses a camera as its sensor. We believe that as the computing power to do image processing increases and the cost of CCD cameras decreases, this will result in a commercially feasible approach for a multi-DOF mouse. Indeed, "solid state" mice[1] which use CCD cameras to sense relative 2D motion will soon be available to the consumer market [18]. The VideoMouse, however, senses 6DOF, including the tilting sensed by the Rockin'Mouse, rotation of the mouse about its vertical axis, and height up to 1 inch from the mouse pad. To implement the vision algorithms with current computing power and image processing techniques, we currently employ a mouse pad printed with a special 2D grid pattern. We believe that this requirement could be removed with future refinements by using tracking and registration techniques (e.g., [18][22]).

We describe the technology and implementation of the VideoMouse as well as interaction techniques which take advantage of the device's unique capabilities. We address several fundamental issues necessary to provide truly mouse-like interaction, such as the ability to move the VideoMouse such that translation is always relative to the current orientation of the device. We describe rate and position control mappings of the input axes, a two-handed technique where the nonpreferred hand rotates the mouse pad to allow quick inspection of a scene (as opposed to an object within the scene), and a technique to facilitate navigation in large 2D structures such as spreadsheets.

UIST '99. Asheville, NC

[1] These solid state mice differ from previous optical mice [16] because they can sense 2D motion over almost any surface.

Beyond 6DOF sensing, the camera of the VideoMouse itself also affords "lightweight scanning" tasks, such as scanning the title of a document or a barcode to bring up a related electronic document on the computer. Since the camera can detect when it is not on the mouse pad, our software can infer when the VideoMouse is being used in a context that is appropriate for image capture.

The VideoMouse's camera-based technology, extended degrees of freedom, and new interaction techniques represent a significant advance which builds on the approach of the Rockin'Mouse. Indeed, we hope that the present paper can help to stimulate additional experimental study and development of interaction techniques for this class of planar multi-DOF devices.

RELATED WORK

A number of previous works explore rotation and tilt-sensing input devices. Balakrishnan *et al.* [1] describe the Rockin'Mouse input device, which is motivated in part by an insightful analysis of the desirable properties of the mouse, such as ease of acquisition due to stability and effective button integration. They present a formal experiment which shows that using the left-right tilt of the Rockin'Mouse to control Z position results in 30% faster 3D positioning than standard modal techniques with a mouse.

Mackenzie *et al.* introduce the Two-Ball Mouse [17], which senses changes in rotation about its vertical axis. The authors classify the degrees of freedom sensed by a variety of input devices and suggest a hypothetical mouse-like 5 DOF device which combines the capabilities of the Two-Ball Mouse and the Rockin'Mouse. We contribute the VideoMouse as an example of such a device.

Kurtenbach *et al.* describe uses of tablet-based rotation-sensing pucks in a prototype artwork application [15]. Such rotation-sensing pucks are now commercially available as the "Intuos 4D Mouse" from Wacom [23]. A task analysis of artwork applications [6] suggests that the ability to rotate an image while sketching can be important. Fitzmaurice *et al.* [7] use 6DOF "bricks" on an ActiveDesk display surface to provide rotation sensing in a drawing application.

Rekimoto [21] as well as Harrison *et al.* [9] suggest techniques for tilt-sensing handheld displays. Similar techniques might prove useful on desktop displays if an appropriate tilt-sensing input device were available.

Several experimental studies suggest that the integrated degrees of freedom provided by 3D devices can indeed be beneficial for some tasks. An experimental study of the Rockin'Mouse found that users can simultaneously control translation and the left/right tilt axis for 3D positioning [1]. Jacob et al. found that a 3D input device can provide superior performance if the user perceives the control degrees of freedom as an integral attribute, whereas a 2D device performs better if the user perceives the DOF's as separable from one another. Hinckley et al. found that users can perform a precise orientation task about 35% faster with an absolute orientation-sensing device than with a 2D device mapped to control orientation [12].

DEVICE DOF ANALYSIS

A careful analysis of the VideoMouse device characteristics is useful both to distinguish it from previous devices and to better reason about what sorts of interaction techniques may be appropriate for such a device. In this regard, we find MacKenzie et al's [17] analysis of the Two-Ball mouse, Rockin' Mouse, and other devices a useful starting point. Table 1 extends this analysis with a more detailed look at exactly what is sensed along each input dimension (*fig. 2*) for the Two-Ball mouse, Rockin' Mouse, and VideoMouse.

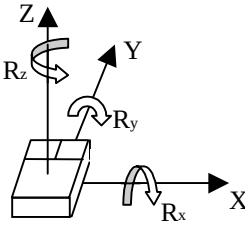

Fig. 2 Device DOF's provided by the VideoMouse.

	Two-Ball Mouse		Rockin' Mouse		Video Mouse	
input DOF	Sensed property	Rate/ Posn[1]	Sensed property	Rate/ posn[1]	Sensed property	Rate/ posn[1]
x	dx	p	x	p	dx	p
y	dy	p	y	p	dy	p
z					z (0-3 cm)	p
R_x			R_x (±60°)[2]	r, p	R_x (±25°)	r, p
R_y			R_y (±60°)[2]	r, p	R_y (±20°)	r, p
R_z	dR_z	p			R_z (360°)	p

Table 1 : A comparison of the sensed properties for each input DOF for several multi-DOF mice.

[1] ***Rate/posn**: This indicates if the sensed DOF property is suitable for **r**ate mode, **p**osition mode, or both **(r, p)**.*

[2] *±60° is the specified tilt range for the Wacom tablet technology used by the Rockin'Mouse.*

An interesting property shared by the Rockin'Mouse and VideoMouse for the R_x (tipping the device forward or back) and R_y (tilting the device left or right) axes is that they can be intuitively mapped to either a position control mode or a rate (velocity) control mode. Since the devices can move through a significant range of degrees (unlike a force-sensing joystick, for example), they are suitable for position mode; an example of such a mapping is the R_y tilting to Z height mapping used by Balakrishnan *et al.* [1] for a 3D positioning task. These input axes are also suitable for rate mode because each axis has a natural center point (the flat spot) and gravity naturally pulls the device back to this center point, both of which are desirable for rate control [24]. Note also that a clutching mechanism may be needed to extend the range of the tilt input when in position mode but is not needed when in rate mode.

The VideoMouse can provide absolute R_z rotation in a full 360 degree range. Because R_z has no self-centering

property, our design intuition was that rate control would probably not work well for this axis; we return to this issue when we discuss 3D manipulation techniques. Since the Two-Ball Mouse bases its rotation sensing on a (dx, dy) motion vector from each roller ball, it can only sense changes in rotation.

The VideoMouse is the only one of these devices that senses height, albeit in a small range of about 1 inch. Our current prototype uses a fixed-focus camera, which limits our ability to resolve the grid at a distance. Note that the sensed height is absolute, but only increases from the device resting state. This provides an excellent match for certain actions, such as making an animated character jump, or significantly, the lifting gesture used to clutch a mouse.

BASIC INTERACTION ISSUES AND FUNCTIONALITY

Traditional mice have a number of compelling properties which have made them the dominant device for 2D desktop interaction. For example, when the mouse is released, it remains in a stable state, ready for reuse. Also, the mouse reclutching mechanism – lifting the mouse and putting it back down elsewhere to extend its range of motion – is simple and effective [1]. We found that it was necessary to address several fundamental device properties of this sort with the VideoMouse to ensure that it would provide mouse-like interaction qualities. Of course, we also needed to implement a robust real-time tracking algorithm to calculate the device motion and orientation information.

Mouse Reclutching Gesture

Unlike a traditional mouse, the VideoMouse does not have a built-in mechanical reclutching gesture. By using the mouse height and tilt angle sensing, however, we can simulate mouse reclutching. Note that one cannot directly use the height parameter to reclutch: when the mouse is tilted but still in contact with the mouse pad, from the user's perspective he or she has not "lifted" the mouse even though the sensed height of the camera above the mouse pad has increased. Instead, we calculate a maximum and a minimum nonlinear height threshold function based on the current tilt angles (R_x and R_y). If the height exceeds the maximum threshold, the device is considered o*ut of proximity*. If the device is already out of proximity and the sensed height falls below the minimum threshold, the device is considered back *in proximity*. Separate minimum and maximum thresholds are used so that slight noise in the height sensing (or from hand tremor) will not result in unstable switching between the two states.

It is also possible to provide this functionality with a hardware switch that senses when the device base makes contact with a solid surface [1]. Our approach uses the sensor information to do this in software.

Motion in Local Coordinates

An important property of mechanical "roller ball" mice is that they always sense device motion relative to the current local orientation of the mouse. This is not true for many optical mice that utilize special mouse pads, because they can only sense device motion relative to the rectilinear coordinate system of the mouse pad. Rather than moving one's hand at whatever orientation seems natural and comfortable, users are instead forced to match hand motions and the orientation of the mouse to the mouse pad.

We soon realized that this was an issue for the VideoMouse as well, since the device does report its motion and current orientation with respect to the grid pattern. But because the device senses absolute rotation, unlike previous optical mice, we can correct for this to provide a mouse that responds *as if* it senses motion relative to its local orientation. Each motion sample is rotated by the current device rotation R_z and added to the previous cursor position. Because absolute rotation is sensed, cursor motion that matches hand motion is always maintained.

Although this is a straightforward observation and easy to implement, this technique results in a major improvement to the usability and comfort of the device for 2D translational movements.

Stability

We were concerned the curved base of the VideoMouse might result in unintended tilting of the device, causing poor 2D positioning. However, in practice we find that the flat spot (as first introduced by the Rockin'Mouse) is quite effective in maintaining the upright posture of the mouse, while still allowing the device to be tilted easily when needed. Also, if the device is released while tilted, it quickly rights itself, so that it is always ready for reuse.

Video Processing

Camera Hardware

The VideoMouse uses a 320x240 CCD camera. Six red light emitting diodes mounted inside the mouse provide illumination. We chose red LED's to match the quantum (best) efficiency of the CCD camera. The video signal is carried to a PC, digitized with an Osprey video capture card, and processed to determine the 6DOF camera pose. The pose information is transmitted to a second PC which renders the 3D graphics scene.

Pose Estimation

Pose estimation (determining the 3D position and orientation of a camera based on the 2D location of known markers) is a well studied problem in computer vision [5]. Since we are estimating 6 DOF, we must find at least three markers (each marker provides two independent pieces of information). More can be used for redundancy. For the mouse pad grid, we can only use planar configurations of markers. Since we want the processing to be relatively straightforward, we use a grid of dots sufficient to encode an absolute up vector and changes in position.

Our technique is based on finding the vanishing points of the parallel lines in the grid. If the calibration pattern were a cube, then we could directly calculate the focal length and rotation of the camera [4]. However, with only a planar grid pattern, we must determine the focal length from at least three different views of the pattern [28].

Our pose estimation algorithm is based on a grid pattern of black dots (*fig. 3*). The *centroid* of each circle can be computed very quickly, and gives a precise location. Once the dot centers have been computed, we connect each dot to its four nearest neighbors and use the line segments connecting nearest neighbors (*edgels*) to extract the grid pose as follows:

1. *Correct for radial lens distortion.* We estimate the lens distortion parameters using a *plumb line method* [8]. When correcting edgels, we re-compute their locations (centers) and orientations.
2. *Extract lines.* We use a Hough transform [2] to vote for lines through the edges.
3. *Extract vanishing point.* We use a generalized Hough transform to vote for vanishing points associated with pairs of nearby lines. Then we go back to the original lines to refine the vanishing point estimates by minimizing $\Sigma_i(\boldsymbol{l}_i^T\boldsymbol{v})^2$, where $\boldsymbol{l}_i$ is the line equation in homogeneous coordinates, and $\boldsymbol{v}$ is the vanishing point in homogeneous coordinates. This can be found as the minimum eigenvector of the symmetric 3×3 matrix $\Sigma_i(\boldsymbol{l}_i\boldsymbol{l}_i^T)$.
4. *Compute the rotation matrix.* We use the best two vanishing point estimates to compute the rotation matrix $\boldsymbol{R}$. The *z*-axis is the cross product of the two vanishing point vectors. Perpendicular *x*- and *y*-axes are found using weighted averages of the vanishing-point vectors and their cross products with the *z*-axis.
5. *Find the phase (2D motion).* The inverse rotation matrix is used to rotate the grid to find a canonical grid (of horizontal and vertical lines) in the projective plane. The *x*-coordinates of the centers of the vertical lines and the *y*-coordinates of the centers of the horizontal lines are used to find the median spacing between grid lines. We then use the horizontal and vertical lines closest to the origin to find the phase of the grid relative to the origin using the median spacing as a frequency.
6. *Estimate the height.* The median spacing between grid lines calculated above is compared with the median spacing found when the VideoMouse is at rest to find the current distance to the mouse pad. We estimate the height by multiplying this distance by the r_{33} component of the rotation matrix.

We put white mini-dots (*holes*) inside selected (enlarged) black dots (*marker dots*) to enable phase computation (2-D motion) over a wider range. After grid lines have been found, lines containing marker dots are identified and the phases of these lines are used to get a position estimate that allows faster mouse motion. For example, given our current sampling rate of 30 Hz, a grid with dots spaced 0.1" apart and marker dots in every third row and column is theoretically limited to motions slower than 4.5 in/s.

By placing the holes slightly higher than the centers of the marker dots, we are able to compute an estimate of the "up" direction when we find the dots. This "up" direction is used to make sure that the *x*- and *y*-axes are oriented correctly when the rotation matrix is computed.

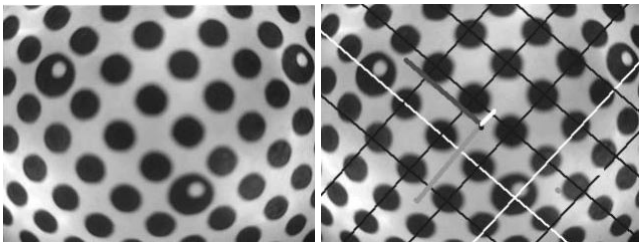

Fig. 3 *Left*: Raw input image showing the 2D grid pattern. *Right*: Vision algorithm extraction of grid orientation.

Revised Grid Design

In practice, the above grid design does not allow mouse motions as fast as we would like, so we added more holes to the marker dots to encode 3-bit row and column coordinates. When large dots with holes are found, the largest hole is used to encode orientation information, as in the previous design. The other holes can fall in six possible locations relative to the centroids of the dot and the largest hole (*fig. 4*). The presence or absence of a hole encodes one bit of row or column position information. This allows mouse motions eight times as fast as the previous design.

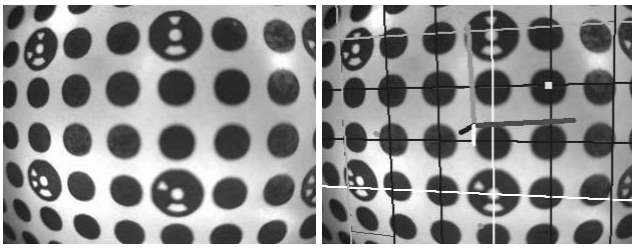

Fig. 4 The revised grid encodes position information in the large dots for better tracking of rapid 2D mouse motions.

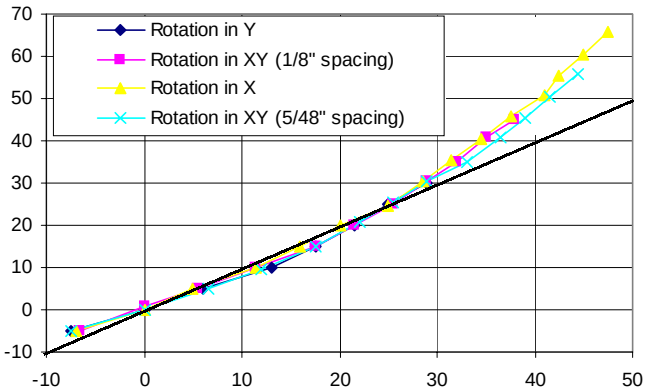

Fig. 5 Actual angle vs. computed angle for the vision tracker. The dotted line shows the ideal response. Angle estimates are fairly accurate between the -20° and 20° range used in practice. Estimates are consistent over different grid orientations and spacings

We used a mechanical device to hold the VideoMouse steady so that we could find the mean and standard deviation of a series of 1000 angle estimates. This device

allowed us to rotate the mouse from side-to-side and measure the angle of rotation to within two degrees. We placed grids under the mouse at different orientations and compared measured angles with computed angles. The angles measured with the video mouse were fairly accurate between -20° and 20° and were consistent over different orientations and grid spacings (*fig. 5*).

The estimates with 45° grid orientations were noisier than with 0° and 90° grid orientations. Most of the standard deviations of the angle estimates were less than .1 degrees, and all standard deviations that we measured were less than .3 degrees, but there are still some uncommon position and orientation combinations for which the code has trouble making angle estimates. While this performance demonstrates the feasibility of our implementation approach, further refinements would be necessary for a commercial version of the device.

Limitations of the Current VideoMouse Prototype

Although the VideoMouse represents a promising technology, we are working to further improve several aspects of the device. Our current set-up with two separate PC's introduces approximately 150-200ms end-to-end lag in the system. We believe that much of this lag will be eliminated when everything is integrated in a single machine. Also, we currently sample the grid at 30 Hz which is slower than we would like. A higher sampling rate would allow a simpler grid design to track rapid hand motions, and would help further reduce lag.

Our new grid design allows significantly faster 2D translation with the mouse, but still has some bugs which can occasionally introduce noise or sudden jumps in the orientation, especially for 45° orientations of the R_z axis. This can also result in inaccurate estimates for the Z height.

Our current curved base for the VideoMouse is hand-made and thus not easily described by a simple analytic equation. When combined with the sometimes erroneous Z height of the new grid design, this can result in erroneous detection of tilting or the *out of proximity* state for the mouse. However, we believe that when we resolve these technical details, software detection of these states will work reliably.

3D OBJECT MANIPULATION WITH THE VIDEO MOUSE

With appropriate interaction techniques, the additional degrees-of-freedom provided by the VideoMouse can support many 3D object manipulation tasks. Although the VideoMouse is a 6DOF sensing device, for the 3D object manipulation techniques we describe in this section, we use only 5 of the 6DOF for manipulation. Lifting the device (Z axis translation) is always interpreted as the mouse reclutching gesture. This style of usage is most similar to the traditional mouse: no matter what the user is doing with the device or what the mode of the software, lifting the device can be depended upon as a gesture to separate device motion from the virtual object or cursor motion. Nonetheless, the VideoMouse sensor does provide the 6th degree of freedom if it is needed for height manipulation. This functionality could be made available if the user held down a keyboard modifier key, for example.

In this section, we assume that a keyboard modifier, the Ctrl key, is used as a clutch, so that holding down the Ctrl key while moving the mouse manipulates the selected object in 5DOF. Releasing the Ctrl key drops the object at its current position and orientation and allows the user to once again move the 2D cursor. We also support clicking and holding the left mouse button as a clutch for 3D object manipulation; this method was actually preferred by most of our test users for its familiarity and simplicity. One advantage of using a keyboard modifier, however, is that it separates the clutch from the hand controlling the device, which allows better fine motor control.

As a final note, we intersperse observations and comments from our usability testing throughout our description of the interaction techniques. Test users tried out the various object manipulation features of the VideoMouse in a simple scene with a ground plane and two cubes. Participants were asked to perform example object manipulation tasks (such as "Move the small cube so that it aligns with the large cube"); these 3D manipulation tasks were interleaved with standard 2D cursor control to select the object to manipulate, for example. Seven test users participated; two users described themselves as "3D gamers."

Rotation About the Vertical Axis

By clicking and dragging an object without tilting the VideoMouse, our software can readily support a 3DOF manipulation mode for 2D translation plus in-plane rotation about the vertical (R_z) rotation axis (*see fig. 2*). Since the hand may rotate at the wrist, elbow, or shoulder as it translates, moving the mouse typically also results in some rotation. Thus interaction techniques for a rotation-sensing mouse should either be designed with a certain amount of forgiveness for unintended rotations, or with a clutching mechanism that allows the user to control whether or not device rotation is used.

Absolute Mode for R_z Rotation

An absolute mapping of the VideoMouse's R_z rotation to the virtual object rotation works well for this axis. During user tests, we observed that with an absolute mapping any unintended rotation that is introduced as the user translates an object can be quickly corrected with a final ballistic rotation of the mouse. We also observed that many users would pinch or twist the mouse in their finger tips to allow dexterous manipulation of the rotation across a wider range of physical movement.

However, a direct ratio of one Control unit to one Display unit (1:1 C:D ratio) is limiting because it is difficult to rotate one's mouse across the full 360 range sensed by the VideoMouse. The R_z rotation is sensed precisely enough that a relatively high C:D ratio can allow an extended range of motion that overcomes the biomechanical limits of the hand while still allowing precise rotation. Our software currently defaults to a C:D ratio of 1:4, but during our usability testing we observed that subjects could work

effectively with settings ranging from 1:2 to 1:5. The "optimal" setting likely depends on individual preferences as well as the specific task at hand.

Rate Mode for R_z Rotation

We also experimented with using a rate mode for rotation about the vertical axis, where deflection of the device from the start of the rotation gesture (when the Ctrl key is pressed) controls the velocity of rotation rather than the absolute rotation angle itself. The rate mapping seems to suffer fundamental limitations because (1) this axis of the device has no inherent self-centering property, which is desirable for rate control [24]; and (2) because as one moves the mouse some rotation is inevitably introduced– thus causing the object to start spinning unexpectedly. Unlike the absolute mode for rotation, such unintentional spinning cannot be corrected in a single ballistic rotation of the mouse, but rather requires a series of gestures to stop the object from spinning, reverse its rotation, and then recenter the mouse to its "zero" orientation.

Nonetheless, some users did prefer the rate mode for rotating the object. One advantage of using a rate mode for R_z is that a reclutching mechanism is not necessarily needed. By rotating and holding the device, the virtual object will continue to spin until it approaches the desired rotation. With an absolute mapping (times a gain factor), when a physical limit in the rotation is reached, the user must "ratchet" the object by releasing the Ctrl key (or by lifting the mouse) to drop it, reset the orientation of the device, and then hold down the Ctrl key to rotate the object some more. The subjects who preferred the rate mode also tended to find such ratcheting motions less intuitive.

The velocity for a given angle is calculated with a nonlinear mapping [24]. A small dead band of ±2° is provided so that the rotation can be brought to a definite stop. The resulting equation is:

$$dR_z = \mathbf{sgn}(R_z) * K * \mathbf{max}(||R_z|| - R_{zmin}, 0)^{\alpha}$$

where dR_z is the calculated velocity, K is the control gain, R_{zmin} is the size of the dead band, and α is the nonlinear parameter. The *sgn* function multiples by the sign of R_z to keep dR_z in the same direction as the rotation.

Tilting the VideoMouse

We use the tilt degrees of freedom (R_x, or tilting forward and back, and R_y, or tilting left and right) to control the other two degrees of freedom of object orientation. A rate control is much more effective for the tilt axes than for the R_z rotation axis because the device does naturally return to its flat spot, where the rates are mapped to zero. Tilting with a rate control allows the user to either quickly spin the object with a large tilt, or to very precisely nudge the rate with a subtle tilting of the hand. The tilt velocity is calculated with the equation shown above, again with a dead band of ±2° for each axis. This dead band prevents a slight accidental tilting of the device, or inaccuracies in the tilt data, from affecting the virtual object.

Note that this interesting combination of rate control for R_x and R_y and absolute rotation control for R_z does not seem to cause any inherent problems. Indeed, this combination helps to circumvent one of the primary objections to rate control – the inability to rapidly rotate an object or the scene to see things from another viewpoint, and just as quickly return the original orientation by "muscle memory" [1]. Since the R_z axis does provide an intuitive absolute mapping, one can quickly flip between widely separated rotations along this axis. However, it is true that movements along the rate-controlled axes could not be immediately reversed in this manner.

We did observe a couple of problems with tilting during our usability testing. First, tilting forward can sometimes result in an accidental mouse button click. Our software can easily ignore most such accidental clicks (because we can detect that the user was tilting the mouse when the click occurred), but the user still *thinks* that he or she has made a mistake because the "clicking" feedback from the button gives the impression that an action has been initiated. A second and more significant problem is that for a right-hander, it is easier to tilt the mouse to the right than to the left. The VideoMouse is adapted from the shell of the Microsoft IntelliMouse Pro, which has a slight slope to the right to encourage a neutral hand posture. We chose this mouse shell because it does make it easier to tilt the hand to the left than if the hand were held flat on the desk surface[2]. Our sense is that increasing this slope may be helpful for tilting, although we have not yet constructed such a prototype. A number of other ergonomic refinements may be possible if one designed a mouse from the ground up to afford tilting motions.

Interactive Audio Feedback

We provide interactive audio feedback proportional to the rate of tilting. The impression is similar to that of a race car accelerating. Separate acceleration sounds are mixed for each tilt axis. This gives the user excellent feedback for exactly how much of each tilt axis is being engaged, even if the object is rotating very slowly and thus perhaps difficult to notice visually. We also provide distinctive "clicks" when a rate-based tilt first begins as well as when the VideoMouse returns to its flat spot. Our sense is that this feedback is both fun and quite useful to help augment and reinforce hand-eye coordination. Unfortunately, however, the audio feedback was not implemented at the time of our usability testing so we do not yet have any reactions from end-users to report, so while the audio seems useful its benefit remains unproven.

In general, a number of other qualities of the system are sonified. For example, dragging the objects (translating in XY) creates a soft sliding noise proportional to the speed of movement. We also provide audio feedback for lifting up or putting back down the VideoMouse; unlike a mechanical

[2] The creators of the Rockin' Mouse also noticed in one device design variation that a "neutral posture" slope seemed helpful [Ravin Balakrishnan, personal communication].

mouse, which makes a noise as it is lifted, lifting the VideoMouse is almost perfectly silent without this feedback. Similarly, the user hears a thudding noise when the VideoMouse reaches the edge of the grid pattern; this makes the user aware of this error condition without having to visually monitor where the mouse is on the mouse pad.

Absolute Mode for Tilting

We implemented an absolute mode for controlling object orientation with the tilt axes, but did not test it with users. The range of rotation that is sensed (and that is comfortable ergonomically) is limited for these axes. Also, lifting the mouse is less effective as a reclutching gesture when tilting the mouse. Thus, a rate mapping seems appropriate for specifying an arbitrary orientation with the current tilt-sensing capabilities of the VideoMouse. Nonetheless, an absolute mapping for the tilt axes can be extremely useful for some tasks (e.g., [1]). We will present another such example later in this paper. Indeed, we see the flexibility of the device to support either rate or absolute mode mappings of the tilt axes from software as one of its strengths.

Device Ergonomics

Obviously, a device that requires extreme wrist flexion would raise significant repetitive strain injury concerns. While we do not believe that planar multi-DOF devices necessarily engender such difficulties, this is a legitimate concern which needs further study. However, our design experience with the VideoMouse suggests the following observations that can help to address ergonomic issues.

First, the 6DOF manipulation literature suggests (and indeed, we believe it is vital) that the device should encourage a finger-tip grip for 3D interaction, rather than a whole-hand grip which mandates wrist flexion. Zhai *et al.* [26] report that the bandwidth of 6 DOF manipulation is maximized when the fingertips participate in manipulation. We believe a mouse design with a smaller, more circular shape and a shorter vertical profile than our current prototype would be helpful in this regard. Furthermore, using the fingertips of both hands encourages comfortable and dexterous 6DOF manipulation [10]. One technique for using both hands to aid manipulation with the VideoMouse is described in the next section.

Second, we believe the degree of tilt or rotation is not so much the key issue as is the hand posture and muscle groups required to orient the mouse. Encouraging a finger-tip grip may help to extend the range of motion for the device while also making it comfortable and pleasant to use. Failing this, if rate control techniques are used for the tilt axes, the angular displacement of the device can be limited to a range of approximately ±5° to 10° for each axis. If subsequent study finds that rate controls really are the best match for the planar multi-DOF axes, the hardware could be designed to physically limit tilting to a small range; we have not yet implemented this as our intuition is that positional mappings of the tilt axes can be quite useful, and furthermore we wanted our prototype device to allow as general-purpose postural sensing as possible.

Our sense is that these issues are subtle and difficult to get right, and given that the VideoMouse prototype adapts an existing mouse shell, the device form-factor could probably be improved by redesigning and reevaluating the device ergonomics from the ground up.

TWO-HANDED ROTATION USING THE MOUSE PAD

Although we initially assumed that one would use the rotation-sensing capabilities of the VideoMouse by rotating the mouse itself, we soon realized that one could just as easily rotate the mouse pad instead. This affords a two-handed interaction technique where the user can rotate the pad with the nonpreferred hand while either holding the VideoMouse still with the preferred hand, or simultaneously counter-rotating the mouse to extend the continuous range of rotation beyond the biomechanical limits of the preferred hand acting alone.

By sensing such nonpreferred-hand use of the mouse pad via touch sensing [11], the mouse pad becomes a "prop" [10] for rotating the scene or ground plane, while rotating the mouse by itself manipulates an object within the scene (*fig. 6*). This gives the user a very lightweight and direct way to inspect the scene from multiple vantage points (and other uses can be envisioned as well [6]). We also reduce the rotational gain factor (to a 1:2 C:D ratio instead of the default 1:4) so that the user may rotate the scene slowly with the nonpreferred hand or rapidly by simultaneously counter-rotating the mouse hand. Users found the technique very intuitive and easy to master.

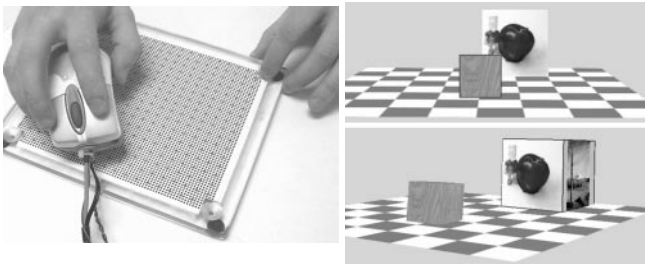

Fig. 6 Two-handed rotation of the mouse pad allows rotation of the entire scene. To resolve the depth ambiguity seen here in the top right image, the user has rotated the ground plane to a different viewpoint (bottom right).

The mouse pad has a finger-sized recess located in each corner of the pad, and we can sense when the user puts his or her finger in one of these recesses to rotate the pad. The user can also to choose to move the mouse pad without triggering the touch sensors by simply grabbing it at any other point. This makes the gesture of grabbing the mouse pad (to rotate the scene) intentional, yet very easy to do. The mouse pad has a raised pivot point which gives it a slight affinity to rotate about its center like a turntable, yet without feeling "wobbly." Our current implementation of the touch-sensing mouse pad requires a wire to tether it to our touch-sensing circuitry. However, we believe it may be possible to build a wireless version using the capacitive coupling techniques described by Zimmerman *et al.* [29].

We currently use only the rotation of the mouse, and ignore any cursor movement. Some change in the cursor position

accompanies the rotation (since it does not occur about the center of rotation of the camera itself). Nonetheless, it might be interesting to map significant forward/back motion of the mouse to control the distance of the camera from the scene, for example.

We should also note that regardless of the mouse pad orientation, the cursor motion as one translates the mouse always corresponds to the user's hand motion. Our technique for mapping mouse motion to the local orientation of the mouse automatically has this effect, even when the mouse pad is upside down, for example.

Kinesthetic Correspondence with Mouse Pad Rotation

Because there are two moving objects (the mouse pad and the mouse), and only one rotation sensor, we know only the relative rotation between the two objects; we do not know for certain which object the user is actually rotating. We initially had some concern because this can result in a situation where the user will rotate the mouse or the mouse pad in one direction, while the objects on the screen will rotate in the other. Such a loss of *kinesthetic correspondence* is generally considered non-intuitive [3].

To investigate this issue, we implemented an option so that the rotation of the ground plane could either be (1) *in correspondence* with the rotation of the mouse pad, assuming the user is moving the pad; or (2) *counter-rotated* to match the rotation of the mouse, assuming that the user holds the mouse pad still and moves only the mouse (if the user were to hold the mouse still and move the mouse pad, the scene would then rotate in the *opposite* direction of the mouse pad rotation with this second option).

When we tried each of these options during user testing, remarkably most users did not even notice if the scene rotated backwards when they rotated the mouse pad. All four test users found either direction of rotation to be intuitive and most were surprised when we pointed out the difference.

One user who did notice this had a tendency to hold the mouse pad still, and rotate the mouse instead. This user felt that the rotation was a bit more natural when it corresponded to the mouse rotation. For this reason, our current default is option (2), *counter-rotation*. Most users who do rotate the mouse pad don't notice that the scene spins in the opposite direction, while users who tend to rotate just the mouse may find this option more natural. We believe that, similar to the old debate about the "correct" direction of scrolling when dragging a scroll bar, users can adopt either a mental model of "the mouse pad is the floor" or "the mouse pad controls a camera spinning around the floor" with equal ease.

SPREADSHEET NAVIGATION

We have also experimented with using the tilting capabilities of the VideoMouse to aid navigation of spreadsheets, which serves as a concrete example of navigation in a large 2D space. Navigating such spaces with standard scroll bars is quite tedious. Not only do scroll bars control only one dimension of panning at a time, but also a distant destination for navigation may be beyond the edges of the screen, and thus invisible to the user. The user must either remember where other information is in the spreadsheet relative to their current location, or search for the desired destination while scrolling. Similar issues also arise with zooming user interfaces [19].

Another approach is to use a menu to change the magnification of the document so that more area can be seen, select a new region, and then revert the magnification to the original setting, but this also is a tedious process involving many steps which interrupt the user's workflow. Some mice with wheels can support this functionality by holding down *Ctrl* and rolling the wheel to zoom, but the navigation occurs in discontinuous steps, which can cause the user to get disoriented, and is still disjoint from the mechanism to select a new region.

We use tilting the mouse forward or back as a distinct gesture to zoom out to an overview of the spreadsheet (*fig. 7*). This can be combined with normal 2D motion of the device to move a semi-transparent selection region. Releasing the mouse button then flies the user into the new region of the spreadsheet indicated by the selection region.

The tilt angle is scaled by a gain factor to allow continuous control of the height of the camera above the spreadsheet. This is an example of a absolute mode for mapping the tilt data to height. In that regard, this mapping is similar to the height control described by Balakrishnan *et al.* [1], but the

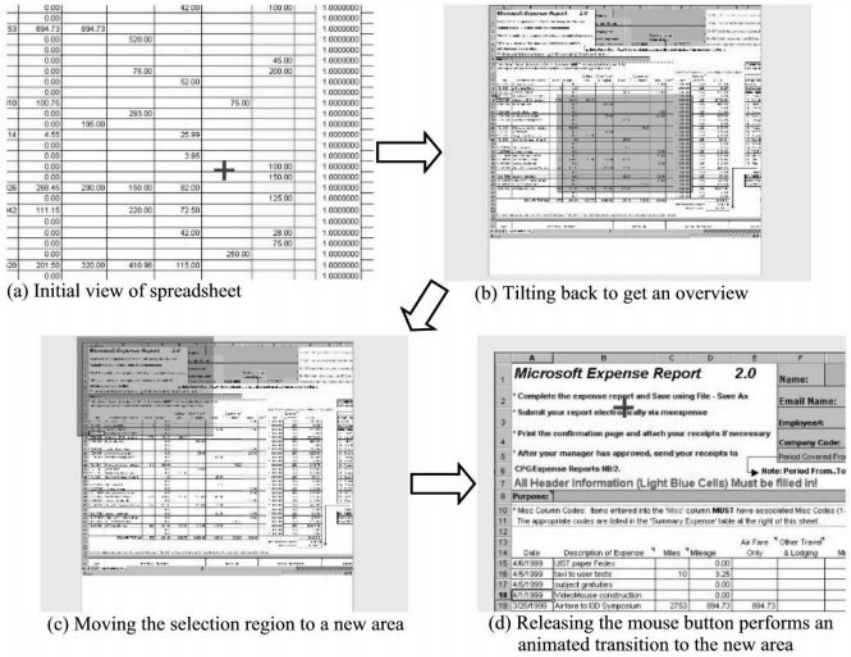

Fig. 7 Several snapshots from a sequence of the user navigating through the spreadsheet.

interaction metaphor is different. Our technique currently does not distinguish between tilting the mouse forward and tilting the mouse back; however, such a distinction might be used to extend the technique to allow both zooming in for more detail as well as zooming out for an overview.

This technique is related to Rekimoto's technique for navigating maps with a tilting display [21]. For desktop usage, a metaphor where the information remains parallel to the screen (rather than tilting away in the distance as demonstrated by Rekimoto) seems more natural. Also, our technique allows the user to combine 2D mouse motion with tilting to simultaneously control the height above the document and the position of the selection region.

Quickly tilting the device up and then back down is also used as a discrete gesture to "show more" of the spreadsheet. This gesture backs out to a fixed overview, and then the user can move the mouse around (while flat on the desktop) to position the selection region. We believe that it would be useful to allow a series of such gestures to continue backing out to higher and higher-level overviews, but we have not yet implemented this.

Test users were very enthusiastic about this capability, but had mixed reactions to using our current absolute function of forward/back tilt to control this action. We plan to explore additional mappings of the VideoMouse degrees-of-freedom to control this functionality. Nonetheless, test users were able to use the technique to quickly navigate to distant areas of the spreadsheet, and felt strongly that the technique makes this action much easier.

LIGHTWEIGHT SCANNING TASKS

We believe the camera of the VideoMouse could help to enable lightweight scanning tasks, which are short, almost trivial scanning tasks that may need to be frequently interleaved with standard GUI interaction. Such tasks can occur naturally when the user works with paper documents that have electronic counterparts [14][20]. An example would be the processing of paper invoices that also exist as electronic records in a database.

The field-of-view of the VideoMouse is too small to make it useful for scanning in large images or entire pages of text. But it is useful for quick scanning tasks that would be too cumbersome if the user had to put down the mouse to switch to a handheld scanner. For example, users could scan the title of a document, and OCR software could be used to trigger a query for related electronic documents.

The VideoMouse has a physical registration marker on its side to assist lining up the camera with the desired text on the piece of paper. We observed that it is easier for users to use this physical registration mark to guide hand motion across the text than it is to use the live video window. One minor problem is that the mouse still occludes the very beginning of the line of text, so the video window does help the user to position the mouse at the beginning of the line.

The VideoMouse pose estimation algorithm calculates a confidence measure which we use to sense when the mouse has been removed from our special mouse pad. Thus, the system knows when the mouse is being used in a context appropriate for scanning, so clicking and dragging the mouse in this case could scan in an image instead of issuing normal mouse clicks to widgets on the screen.

When the user finishes scanning a video sequence, we *stitch* together the individual images in a way that resembles hand-held scanners. We can register successive input frames to create a single *mosaic* [22]. If we assume the user is keeping the mouse flat on the page, after undoing the radial distortion of the camera, we only need to estimate 3 DOF (2-D translation and in-plane rotation) when building the mosaic. Fig. 8 shows a single, uncorrected, input image and the resulting mosaic computed from the entire video sequence. In order to prevent successive errors in rotation estimation from accumulating, we could also try to estimate the absolute orientation of each input image (e.g., by Houghing the input edgels), but we have not yet implemented this.

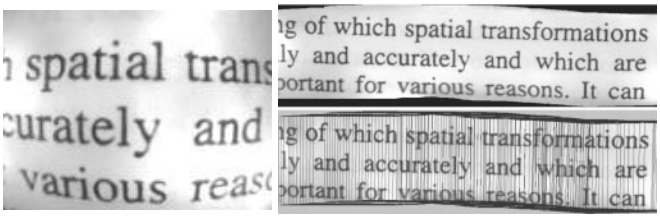

Fig. 8 *Left*: A single input image. The camera senses that it is no longer looking at the grid. *Top Right:* Results of the image mosaic algorithm. *Bottom Right:* Contributions of individual images to the mosaic.

We have implemented camera sensing of the mouse context and the image processing necessary to reassemble the resulting video sequence. This demonstrates the ability to intelligently capture short scanned sequences. We have not yet implemented a mechanism to bring up the live video on the user's primary monitor (a secondary monitor is currently used). We also need to integrate optical character recognition software with our system to allow the extraction of actual text. Nonetheless, we believe the implemented system and design ideas clearly demonstrate the potential value of having simple image capture facilities tightly integrated with the keyboard-mouse user interface.

CONCLUSIONS AND FUTURE WORK

Additional study of the class of planar multi-DOF devices is needed. Experimental studies might include assessing the overall task performance for both 2D cursor movement and 3D manipulation, as well as measuring users' ability to coordinate all 6DOF [25]. The device capabilities also call for improved design of the physical form factor and careful ergonomic analyses. We are considering designs that move the device center of rotation to the nodal point of the camera, as well as spring-loading of the base to provide feedback for tilting and return-to-center force.

A great deal more work needs to be done on interaction techniques to map the device DOF's to other tasks of interest. For example, we feel that the VideoMouse may enable interesting new 3D navigation techniques. It may

also provide exciting new interaction for 3D computer games or flight simulators. As another example, manipulators or “3D Widgets” to map 2D mouse motion to various parameters for 3D object manipulation have become standard in many 3D applications. A multi-DOF device like the VideoMouse might enable new sets of such manipulators that map the device DOF’s in task-dependent ways. Note that such multi-DOF manipulators could act at a much higher level of abstraction than existing 3D widgets.

This paper has presented a new camera-based input device with multi-DOF sensing and image capture capabilities. We believe that the camera-based technology will result in a practical and commercially feasible extension to current mouse technology. We have discussed a number of interaction techniques for such a device including 3D manipulation, two-handed rotation of the mouse pad, navigation in large 2D structures such as a spreadsheet, and use of the camera for lightweight scanning.

ACKNOWLEGEMENTS

Thanks to Dave Thiel for audio feedback, Dan Robbins for photographs, and Mary Czerwinski for user test assistance.

REFERENCES

1. Balakrishnan, R., Baudel, T., Kurtenbach, G., Fitzmaurice, G., "The Rockin'Mouse: Integral 3D Manipulation on a Plane," CHI'97, 311-318.
2. Ballard, D.H., Brown, C.M., "Computer Vision". 1982, Englewood Cliffs, New Jersey: Prentice-Hall.
3. Britton, E., Lipscomb, J., Pique, M., "Making Nested Rotations Convenient for the User," Computer Graphics, 12 (3): p. 222-227, 1978.
4. Caprille, B., Torre, V., "Using Vanishing Points for Camera Calibration," International Journal of Computer Vision, 4 (2): p. 127-139, 1990.
5. Dementhon, D., Davis, L.S., "Model-based object pose in 25 lines of code," International Journal of Computer Vision, 15 (112): p. 123-141, 1995.
6. Fitzmaurice, G., Balakrishnan, R., Kurtenbach, G., Buxton, B., "An Exploration into Supporting Artwork Orientation in the User Interface," CHI'99.
7. Fitzmaurice, G., Ishii, H., Buxton, W., "Bricks: Laying the Foundations for Graspable User Interfaces," CHI'95, 442-449.
8. Fryer, J., Brown, D., "Lens Distortion for Close-Range Photogrammetry," Photogrammetric and Remote Sensing, 52 (1): p. 51-58, 1986.
9. Harrison, Fishkin, Gujar, Mochon, & Want, "Squeeze Me, Hold Me, Tilt Me! An Exploration of Manipulative User Interfaces," CHI'98, 1998, 17-24.
10. Hinckley, K., Pausch, R., Proffitt, D., Kassell, N., Two-Handed Virtual Manipulation, TOCHI 5 (3) 1998.
11. Hinckley, K., Sinclair, M., "Touch-Sensing Input Devices," To appear in CHI'99.
12. Hinckley, K., Tullio, J., Pausch, R., Proffitt, D., Kassell, N., "Usability Analysis of 3D Rotation Techniques," UIST'97, 1-10.
13. Jacob, R., Sibert, L., McFarlane, D., Mullen, M., Jr., "Integrality and Separability of Input Devices," ACM TOCHI, 1 (1): p. 3-26, 1994.
14. Johnson, W., Jellinek, H., Klotz, L., Rao, R., Card, S., "Bridging the paper and electronic worlds: the paper user interface," INTERCHI'93, 1993, 507-12.
15. Kurtenbach, G., Fitzmaurice, G., Baudel, T., Buxton, B., "The Design of a GUI Paradigm based on Tablets, Two-hands, and Transparency," CHI'97, 35-42.
16. Lyon, R., Haeberli, M., "Designing and testing the optical mouse," VLSI Design, Jan.: p. 20-30, 1982.
17. MacKenzie, I.S., Soukoreff, R.W., Pal, C., "A Two-Ball Mouse Affords Three Degrees of Freedom," CHI'97 Conference Companion, 1997, 303-304.
18. Microsoft IntelliMouse Explorer, www.microsoft. com/presspass/features/1999/04-19mouse.htm , 1999
19. Perlin, K., Fox, D., "Pad: An Alternative Approach to the Computer Interface," SIGGRAPH `93, 1993.
20. Rao, R., Card, S., Johnson, W., Klotz, L., Trigg, R., "Protofoil: Storing and Finding the Information Worker's Paper Documents in an Electronic File Cabinet," CHI'94, 1994, 180-185, 477.
21. Rekimoto, J., "Tilting Operations for Small Screen Interfaces," UIST'96, 167-168.
22. Szeliski, R., Shum, H.-Y., "Creating full view panoramic image mosaics and texture-mapped models," SIGGRAPH'97, 251-258.
23. Wacom Intuos tablet, http://www.wacom.com/ productinfo/ intuos9x12.html, 1999.
24. Zhai, S., "Human Performance Evaluation of Manipulation Schemes in Virtual Environments," Proc. IEEE VRAIS'93, 155-161.
25. Zhai, S., Milgram, P., "Quantifying Coordination in Multiple DOF Movement and Its Application to Evaluating 6 DOF Input Devices," CHI'98, 320-327.
26. Zhai, S., Milgram, P., Buxton, W., "The Effects of Using Fine Muscle Groups in Multiple Degree-of-Freedom Input," CHI'96, 308-315.
27. Zhai, S., Smith, B.A., Selker, T., "Improving Browsing Performance: A study of four input devices for scrolling and pointing tasks," INTERACT'97, 286-292.
28. Zhang, Z., A Flexible New Technique for Camera Calibration. Microsoft Research Technical Report MSR-TR-98-71, 1998.
29. Zimmerman, T., Smith, J. R., Paradiso, J., Allport, D., Gershenfeld, N., "Applying Electric Field Sensing to Human-Computer Interfaces," CHI'95, 280-287.