FlowMouse: A Computer Vision-Based Pointing and Gesture Input Device

Andrew D. Wilson and Edward Cutrell

Microsoft Research
One Microsoft Way, Redmond, WA
awilson@microsoft.com, cutrell@microsoft.com

Abstract. We introduce FlowMouse, a computer vision-based pointing device and gesture input system. FlowMouse uses optical flow techniques to model the motion of the hand and a capacitive touch sensor to enable and disable interaction. By using optical flow rather than a more traditional tracking based method, FlowMouse is exceptionally robust, simple in design, and offers opportunities for fluid gesture-based interaction that go well beyond merely emulating pointing devices such as the mouse. We present a Fitts law study examining pointing performance, and discuss applications of the optical flow field for gesture input.

1 Introduction

Today's computing environments are strongly tied to the availability of a high resolution pointing device, and, more fundamentally, to the notion of a single, discrete two-dimensional cursor. Modern GUIs (graphical user interfaces) combined with devices such as mice and track pads are extremely effective at reducing the richness and variety of human communication down to a single point. While the utility of such devices in today's interfaces cannot be denied, there are opportunities to apply other kinds of sensors to enrich the user experience. For example, video cameras and computer vision techniques may be used to capture many details of human shape and movement [24]. The shape of the hand may be analyzed over time to manipulate an onscreen object in a way analogous to the hand's manipulation of paper on a desk. Such an approach may lead to a faster, more natural, and more fluid style of interaction for certain tasks [10], [28].

The application of video cameras and computer vision techniques as an interface component to today's computing architecture raises many questions. Should the new device be designed to replace the mouse? In what ways might such a device complement the mouse? Will the added functionality of the new device be incorporated into today's computing experience [15], [27] or does the entire interface need to be rethought [16], [26]?

Unfortunately, most computer vision-based user interface systems are poorly suited to the task of emulating the mouse. To begin, the resolution of video-based techniques is far less than today's mouse. While it is difficult to directly compare the

resolution of an optical mouse to a computer vision-based tracking system, we note that a typical optical mouse has a resolution of about 400 dpi and frame rate of 120 Hz. A typical video camera produces 640×480 pixel images at 30 Hz. Assuming a full view of a 9" wide mousing surface, a computer vision tracking algorithm with tracking precision equal to that of the input image yields a resolution of only about 71 dpi.

The lack of spatial and temporal resolution is not the only difficulty. Often with computer vision-based user interfaces, it is difficult to determine the precise relationship between the object being tracked and the resulting output position information. For example, in tracking the hand, which part of the hand should indicate cursor position? If the tip of the finger is chosen, which point is exactly the "tip" of the finger? A lack of agreement between the user and the sensing system on what is being tracked further limits resolution and can be the source of breakdowns in interaction. A number of related works pursue finger tracking approaches to recover absolute finger position [25], [21], [15], [19]. Another approach is to design a handheld prop which is tracked unambiguously [7], [4].

A related issue is that vision-based systems often have difficulty in providing natural ways for the user to enable or disable the device. Mice and trackpads both provide unambiguous and simple mechanisms that require little or no thought on the part of the user. It is important for vision-based systems to adopt mechanisms that are similarly natural and failsafe, or the trouble of unintentional input will outweigh any benefit provided by the interface.

Finally, many vision techniques make strong assumptions about the appearance of the tracked object and the background or are sensitive to lighting conditions. In the case of computer vision systems trained on hands, often much effort is placed on developing models of skin color and hand shape. The *segmentation* problem of separating foreground from background, based on color or otherwise, is very difficult in general. While the field of computer vision has techniques that can address these problems, the resulting systems can be complex and provide no guarantee on robustness.

In this paper we present a pointing device and gesture input system based on computer vision techniques. To capture the motion of the hand, FlowMouse uses optical flow techniques rather than traditional absolute position-based tracking methods, and so avoids many of the difficulties mentioned above. In a laptop configuration, a natural mode switching mechanism is provided by a touch-sensitive strip placed on the mouse button just below the keyboard. The flow computation performed at each point in the input image is roughly analogous to that performed by a typical optical mouse sensor, in which mouse velocity is determined by image correlation between successive captured images. In aggregate these individual motion estimates provide a robust estimate of the relative motion of the hand under the camera. This approach avoids the fragility of absolute tracking techniques as discussed above.

Flow fields are able to express patterns of motion beyond a simple translation of the hand, and in that capability there is opportunity to explore new interaction scenarios while maintaining support for traditional two-dimensional pointing. Our goal is to demonstrate that FlowMouse is a capable pointing device for today's interfaces while outlining its potential to simultaneously support novel interactions that take advantage of the richness and subtlety of human motion.

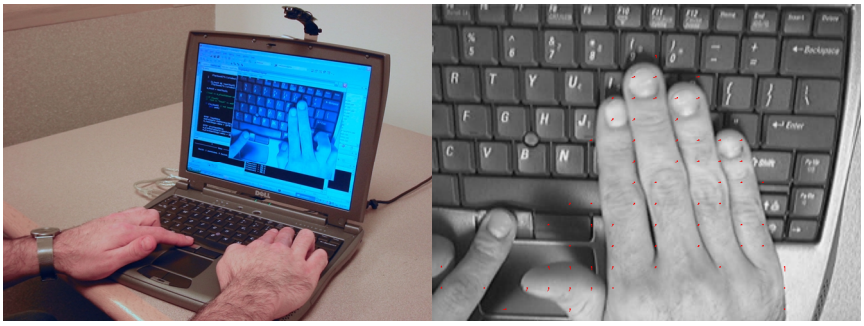

Figure 1 Left: FlowMouse prototype with screen-mounted camera facing down on keyboard and user's hands, and touch sensitive strip on left mouse button. Right: Example image capture from camera.

In the following, we outline the configuration of the device, detail the image processing used, present a Fitts law analysis of FlowMouse pointing performance, and discuss how FlowMouse enables interaction scenarios beyond traditional two-dimensional pointing.

2 FlowMouse

Figure 1 illustrates a FlowMouse prototype installed on a laptop computer. A USB web camera is attached to the top of the display such that the camera image contains a view of the laptop keyboard and the user's hands when they are on the keyboard and mouse buttons. When FlowMouse is enabled, images are acquired from the camera and processed to determine motion information useful for pointing and gesture. In our experiments, we have relied on ambient office lighting for illumination, but we envision that future implementations may include light emitting diodes (LEDs) to be used when ambient light levels are inadequate.

It is important to have a reliable and natural *mode switch* which can be used to enable and disable FlowMouse, such that only intentional gesturing is processed by the system, but also to allow for a "clutching" action that a relative pointing scheme requires [5]. In our prototype, a touch sensor is affixed to the surface of the left mouse button. We chose this placement based on the desire to find a mode switch method that requires very little modification of the user's behavior: we have observed that while moving the cursor using keyboard-integrated devices (e.g., a trackpad or isometric joystick), most users rest their left thumb or left forefinger on the left mouse button so that they are prepared to quickly click the mouse button. We believe that touching the left mouse button is an adequate indicator for using the mouse rather than the keyboard. A similar application of touch sensors is presented in [11] and [22].

The loading of the mode switch on the left mouse button also avoids the problem that many vision-based user interface designs face: if the hand used for positioning is also used to effect a click action, the motion for the click action may be confused with

the motion used for positioning. In this case, the clicking action is likely to bump the cursor off the target just as the user has finished positioning the cursor.

In our prototype, a simple capacitance-based touch sensor relies on a copper electrode taped to the mouse button. Such a sensor can also be placed under the plastic shell of the mouse button itself.

3 Image Processing

While the user touches the left mouse button, FlowMouse sensing is enabled. During this time, grayscale images are acquired from the USB camera attached to the top of the display. These images are then processed in real time to determine the optical flow field corresponding to the motion of the hand under the camera. Optical flow is a standard representation used in computer vision which indicates the direction and magnitude of motion at each point on a regular grid defined on the input image [1], [2].

Part of the goal of FlowMouse is to explore the advantages of optical flow over a more traditional approach based on segmentation of hands against the keyboard and subsequent absolute position tracking processes. Optical flow computations make very few assumptions about the nature of the input images and typically only require that there be sufficient local texture on the moving object. We avoid the difficulties of developing a reliable absolute position-based tracker by only computing simple statistics that summarize the flow field and restrict ourselves to computing relative motion information. As with traditional mice and track pads, a key to the success of this approach is the effectiveness of the clutching mechanism.

During the time the user touches the left mouse button, optical flow is computed from the most recently acquired image and the previous image. There are a number of methods to compute optical flow. Our prototype uses a simple block matching technique in which, for each point (x, y) on a regular grid in the image, the integer vector quantity (dx, dy) is determined such that the image patch centered on (x, y) at time $t-1$ most closely matches the image patch centered on $(x+dx, y+dy)$ at time t. In this calculation, image patches are compared by computing the sum of pixelwise absolute differences (low values indicate close match). For a given patch in the image, we select (dx, dy) that minimizes

$$\sum_{x,y \in \text{patch}} \left| I_{t-1}(x, y) - I_t(x+dx, y+dy) \right| \qquad (1)$$

Our current implementation acquires 640×480 pixel grayscale images at 30Hz. Flow vectors are computed every 32 pixels on a regular grid, yielding a 20×15 flow field. Each of dx and dy are allowed to vary by 6 pixels in either direction on 16×16 pixel patches, and the optimal (dx, dy) for each grid point is found by exhaustive search over this range. The flow field is computed at full frame rate (30Hz) on a 1.1GHz Pentium III Mobile. An example flow field for hand motion is illustrated in Figure 2.

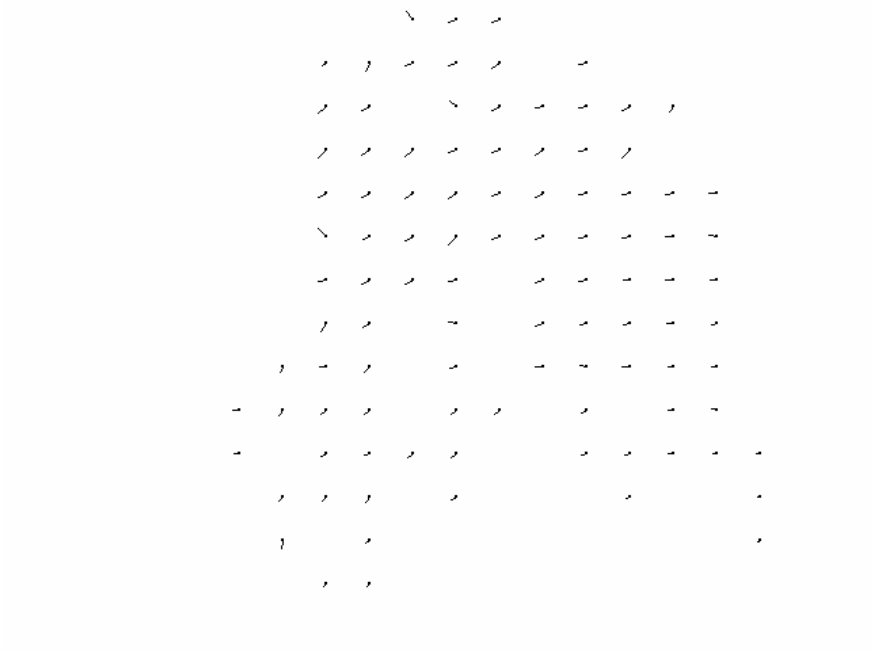

Figure 2 Example optical flow field, with hand undergoing translation left and down.

There are a few details to note about the optical flow computation. First, correct values (dx, dy) are attainable only if there are adequate features such as edges or corners in the image patch under consideration. In practice, it is necessary to determine the merit of the computed (dx, dy) at a patch. We compare the match score corresponding to $(dx, dy) = 0$ against the best score for any (dx, dy) and discard the flow information at this patch if the best score is not significantly better than that corresponding to $(dx, dy) = 0$. This typically avoids the problem of spurious flow vectors computed on regions of the image without adequate texture, such as may be found on the smooth area on the back of the hand.

A second consideration is that this method finds integer values for dx and dy. This would seem to limit the overall precision of motion information derived from the flow, but typically a hand under the camera will generate many valid flow observations. While a single flow observation may be a noisy estimate of motion, when averaged together the collection of flow vectors result in a more stable estimate of motion.

4 FlowMouse as Pointing Device

4.1 Mouse Acceleration Profile

A simple averaging of the nonzero flow field vectors may be used to obtain a grand mean (dx, dy) suitable for cursor movement. In our experience it is necessary to transform this raw velocity to incorporate acceleration such that it is possible to finely position the mouse cursor. Whereas in previous studies such velocity transfer functions are motivated by minimizing the range of motion [14], we adopt an acceleration profile in an attempt to mitigate the lack of resolution of the camera compared to the mouse.

We adopt a sigmoidal (logistic) acceleration profile, where the speed s is computed as the norm of (dx, dy). The acceleration factor scales input (dx, dy) to obtain mouse movement offsets (m_x, m_y) on a display of resolution 1024×768:

$$s = \sqrt{(0.5dx)^2 + (0.7dy)^2} \tag{2}$$

$$(m_x, m_y) = \left(\frac{80}{1 + e^{-(s-4)/0.6}} \right) (dx, dy) \tag{3}$$

where the scaling factors on dx and dy were added to differentially scale movement in the horizontal and vertical directions, respectively, to account for the fact that vertical motion of the hand in the plane of the keyboard appears to be more difficult than horizontal motion. This disparity is probably due to the rotation of the wrist as a component of horizontal movement.

This acceleration profile is more aggressive than the usual profile used in Windows XP [3]. This change reflects the fact that FlowMouse has significantly less sensor resolution than a typical mouse, we require fine positioning of the cursor as well as the ability to move the cursor across the entire screen with little or no clutching.

5 Laboratory User Study

To objectively evaluate the performance of FlowMouse used as a pointing device, we performed a user study using a Fitts Law task. Fitts Law is a standard method for evaluating, optimizing, and studying properties of pointing devices and techniques that is well-accepted by the HCI community [20], [9]. We tested the FlowMouse against the default trackpad included in the laptop the prototype was installed on.

5.1 Hypotheses

Because the FlowMouse technique is quite novel for most users and the prototype is not highly optimized for pointing (as opposed to the trackpad), we expected that the performance of FlowMouse would be significantly worse than the trackpad. Nevertheless, we thought that inexperienced users would be able to complete all trials with little difficulty. In addition, we expected that the performance difference between FlowMouse and the trackpad would be considerably less for users who have a bit more experience. That is, experienced users would show substantially improved performance over novice users.

5.2 Participants

We recruited 6 participants between the ages of 30 and 55 from coworkers. All had extensive experience using trackpads on laptop computers, and none had ever used the FlowMouse. All rated themselves as advanced computer users and had normal or corrected to normal vision with no color blindness. In addition, all were right handed and used the mouse in their right hand. Two other participants who had several hours practice using FlowMouse were recruited to compare the effect of experience on performance.

5.3 Method

The Fitts Law task was administered using a modified version of WinFitts (courtesy of the Dept. of Computer & Information Science, University of Oregon). For each device, participants performed a block of practice trials to familiarize them with the task and device. They then performed a block of trials for that condition. Each block consisted of 2 trials for each of the 12 distance-width combinations at 8 different target angles for a total of 192 trials per block. Error conditions (where a target was missed) were repeated in a random order at the end of the block. The Fitts parameters used in the experiment were: Width: 5, 10, 20 mm; Distance: 20, 40, 80, 160 mm; Angle: 0, 45, 90, 135, 180, 225, 270, 315 degrees from start point. This yielded Fitts index of difficulty measures ranging from 1 to 5.04 bits (according to the formula $ID=log_2(D/W+1)$).

5.4 Results

All data analyses for movement times were performed on the log transformed movement times to normalize the typical skewing associated with response time data. These were converted back to normal time for all figures below to make the results more intelligible. Movement times were first cleaned by removing error trials and outliers (movement times greater than 4 standard deviations larger than the mean for each condition), about 2% of all trials. We collapsed across angle to yield the means of 16

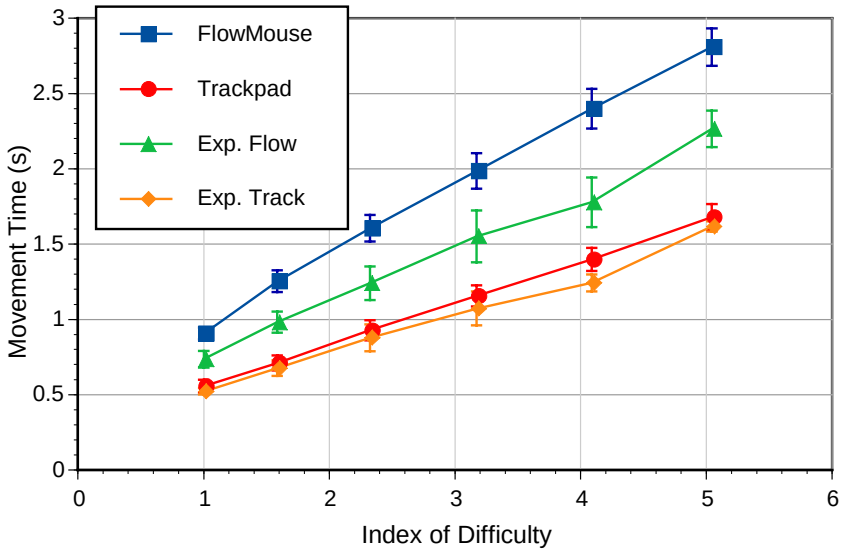

Figure 3 Mean movement time (±SEM) versus Fitts index of difficulty (*ID=log$_2$(D/W +1)*).. Circle & squares represent performance for the 6 novice users of FlowMouse. Triangles & diamonds represent the performance for the 2 experienced users of FlowMouse.

repetitions of each distance-width combination for each participant. The error rate was very low: 1.7% for the trackpad and 2.2% for FlowMouse.

We performed a 2 (Condition) x 4 (Distance) x 3 (Width) Repeated Measures ANOVA on the log-transformed movement data. The typical finding of increased movement time as D increases and W decreases was confirmed (i.e., as the task got more difficult: for D, $F(3, 15) = 709$, $p<<0.01$; for W, $F(2, 10) = 275$, $p<<0.01$). There was also a small interaction between Distance & Width, $F(6, 30)=8.9$, $p<0.01$—as D decreased, the size of the target, W, had a smaller effect on movement time. As hypothesized, there was large difference between conditions; the trackpad was faster than FlowMouse by 700 ms, $F(1, 5) = 555$, $p<<0.01$ (see Table 1).

Because FlowMouse was quite novel for our participants, we were interested in the performance of users with a bit more experience in using the device. Therefore, we looked at two other users who had several hours of practice using FlowMouse. As we hypothesized, a bit of practice resulted in a substantial improvement in performance,

Table 1 Fitts index of performance (IP, the inverse slope of each line in) and the mean movement time (±SEM) for each condition. Note the large improvement in IP for the FlowMouse with a small amount of practice.

	FlowMouse	**Trackpad**	**Exp. Flow**	**Exp. Track**
IP (bits/s)	2.15	3.70	2.76	3.84
Move Time (s)	1.70 (±0.11)	1.00 (±0.07)	1.33 (±0.15)	0.93 (±0.10)

reducing the mean difference in conditions almost in half (only 400 ms difference for the experienced users, see Table 1).

To better characterize FlowMouse, we calculated the Fitts Index of Performance (IP) for FlowMouse and the Trackpad. Fitts Law states that movement time is related to the distance and width of the target being acquired ($MT = a + b \log_2(D/W + 1)$, where a and b are device-dependent empirically derived constants. The inverse slope, $1/b$, is referred to as the Index of Performance (IP) for a given device. This is often used to compare the "bandwidth" of different devices (see [20] and [9] for more details). Figure 3 plots the mean movement time versus ID for each device and participant group (the 6 novice and 2 experienced users). IP was calculated from a linear regression on each line (see Table 1).

Note that the performance on the trackpad is very similar for both novice and experienced FlowMouse users. This is not surprising because both groups were very experienced at using the trackpad. However, a few hours' practice resulted in a substantial improvement for the FlowMouse. This is very encouraging, as it suggests that a combination of device optimization and modest practice could result in performance similar to that of the trackpad.

Participants made several interesting subjective observations after using FlowMouse. All noted that it took a little while to "get the hang" of using it, but after 5 minutes or so, all of them found it quite intuitive. Surprisingly, the most common complaint was not with the FlowMouse *per se*, but the implementation of the touch sensor used to turn it on and off. The sensor was somewhat noisy and participants often had a hard time maintaining contact when they intended to.

Perhaps related to this issue with the touch sensor, the main difficulty for most users was in clutching. Most initially tried to clutch in the same way they would do so with a mouse or trackpad, but because the system tracked the movement of their hand as they clutched, this technique was generally frustrating. Most used the left hand touch sensor to turn off tracking during the clutch motion, but this coordination was somewhat difficult and unintuitive. One novice and both experienced participants adopted a different behavior for clutching. By exploiting the acceleration profile, it was possible to use differential slow and fast movement of the pointing hand to reset the hand position relative to the cursor. This was more intuitive and seemed to result in a generally better performance.

6 Beyond Pointing

For two-dimensional pointing tasks, it is sufficient to collapse the flow field information to a single direction and magnitude by averaging all nonzero points of the flow field. This average indicates the translation of the hand. But flow fields are able to express patterns of motion beyond translation and have been shown to be useful in gesture recognition tasks [8]. Figure 4 shows flow field motions other than a simple translation. We consider a variety of interaction scenarios that are enabled by simple calculations on the flow field:

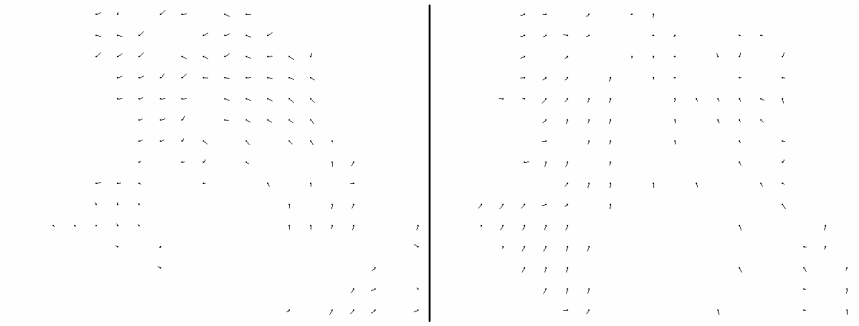

Figure 4 Flow fields represent more than translation. Left: Flow field generated by clockwise rotation of the hand. Note how the flow field vectors lie along circles about the point of rotation. Right: Flow field generated by hand raising over keyboard. Here the flow vectors indicate an expansion about a point.

Freeform rotation, scaling, translation An onscreen object may be simultaneously rotated, scaled and translated in 2D by constructing the transform matrix for a graphics object as the composition of rotation, scaling and translation parameters. In the appendix we present a mathematical framework for computing the simultaneous rotation, scaling and translation of the hand by computations on the flow field. If an application requires it, one or more of the three transforms may be ignored. For example, general object manipulation in a CAD program may make use of all transforms simultaneously, while rotating an onscreen knob may require only rotation information.

Change in hand height to zoom By the same mathematical technique a change in the height of the hand over the keyboard can be detected as a scaling of the flow field. This may be useful for interfaces with zooming functionality, such as mapping programs or other spatial displays.

Tilt By extension of the mathematical framework presented in the appendix, it may be possible to extract the hand's tilting in and out of the plane of the keyboard. This could be useful for "open" or "reveal" gestures, as well as with CAD object manipulation. Tilt could be used to change the attitude of the user's viewpoint in a virtual environment for navigation, including games in which the player controls a vehicle (e.g., airplane). See [4] for related mathematical techniques for extracting object pose in a controller device setting more generally.

Two-dimensional scrolling or panning FlowMouse may be used for pointer based interactions that are complementary to cursor control. For example, the average (dx, dy) may be mapped to two-dimensional scrolling. This panning mode may be triggered by detecting that the user has spread the hand fully, to effect a rather direct analogue of the hand icon found in many drawing and mapping applications.

Scrolling or panning with FlowMouse with the non-dominant hand may complement simultaneous use of the conventional mouse for precise selection tasks [12].

Gesture modulation based on size or shape The approximate size of the moving object under the camera may be determined by counting the number of nonzero points in the flow field. Many of the interactions discussed here may be modulated based on detected size of the moving object. For example, the mouse acceleration profile may be related to object size: smaller motion patterns resulting from the movement of a single finger can yield small amounts of acceleration, while whole hand motion results in the greatest acceleration. This corresponds with the notion that in the real world, gross positioning of an object may be accomplished by whole-hand grasping, while precise positioning may be better accomplished by a gentle nudge with a finger or two.

Simple gesture recognition Strong motion patterns detected over a short duration of time may be set to trigger application commands such as “next”, “previous”, “up”, “down”, “delete”, “dismiss”, “minimize”, “reveal”, etc. [27]. Simple gestures may have their use in short fleeting interactions that do not warrant the burden of fully acquiring the mouse and keyboard, such as in advancing a slide during a presentation, changing the track or volume on media playback, and in other “casual” computing settings.

Marking menus Marking menus typically arrange menu options in a circular fashion around a given point on the screen, like slices of a pie, and so do not require precise cursor positioning along the usual menu items [17]. As such, the coarse relative motion provided by FlowMouse may be well suited to marking menus. The application of marking menus to vision-based UIs is discussed in [18].

Two handed interactions Simple clustering techniques on the position of flow vectors or on their dominant directions of movement (as in [8]) may be used to detect the motion of two hands under the camera. Various two-handed interactions would thus be enabled (see [6] and [12] for examples).

Finally we note that the camera enables a variety of other camera-based interaction scenarios not limited to optical flow. For example, visual tags or barcodes placed on documents on other objects may be recognized [23]. For a fixed camera looking at the keyboard, it would be possible to detect which key the user is about to press and exploit that in some interaction, perhaps in presenting tooltips or status. A camera that can be re-oriented can be used in video conferencing scenarios.

7 Conclusion

FlowMouse is a computer vision-based pointing and gesture input device which relies on optical flow computation where most previous related works uses more fragile object tracking techniques. FlowMouse makes few assumptions regarding the appearance or color of the user's hand, but instead is driven by analysis of the motion patterns indicated in the optical flow field.

The relative motion indicated by the optical flow field may be used for cursor positioning tasks much in the same manner that an optical mouse recovers velocity by image correlation techniques. Our first FlowMouse prototype relies on a touch sensor on the left mouse button for a clutching mechanism. We conducted a Fitts law study which demonstrated that while pointing performance with FlowMouse was significantly worse than with a trackpad, subjects were able to use FlowMouse successfully. Encouragingly, there is some indication that users are able to improve performance dramatically with practice.

The optical flow field derived from hand motion is able to express more than simple translation. We outline a number of interaction scenarios which make use of the rich motion information present in the optical flow field, such as rotation and scaling effects. Furthermore, we note some scenarios where FlowMouse may complement the mouse, such as simultaneously pointing and panning.

In the same way that today's pointing devices are the product of years of engineering refinement, FlowMouse can take advantage of numerous technical improvements to enhance pointing performance. For example, faster cameras are available today, and better optical flow techniques can be used to obtain flow fields of higher quality. However, we believe that FlowMouse and related devices will ultimately make the most impact in providing a channel of input that is richer and more expressive than that of today's pointing devices. FlowMouse in particular is focused at capturing the richness and subtlety of human motion for novel interactions while offering a kind of "backwards compatibility" with today's point-and-click interfaces. By way of extending today's interfaces, rather than replacing them completely, we hope that novel devices such as FlowMouse will find easier paths to adoption and present opportunities for significant user interface innovation.

References

1. Anandan, P., *A Computational Framework and Algorithm for the Measurement of Visual Motion*. International Journal of Computer Vision, 1989. **2**: p. 283-310.
2. Barron, J., D. Fleet, S. Beauchemin, and T. Burkitt. *Performance of Optical Flow Techniques*. in *Computer Vision and Pattern Recognition*. 1992.
3. Bathiche, S., *Pointer Ballistics for Windows XP*, in *http://www.microsoft.com/whdc/device/input/pointer-bal.mspx*. 2002.
4. Bradski, G., V. Eruhimov, S. Molinov, V. Mosyagin, and V. Pisarevsky. *A Video Joystick from a Toy*. in *Proceedings of the 2001 Workshop on Perceptive User Interfaces*. 2001.
5. Buxton, W. *A Three-State Model of Graphical Input*. in *INTERACT '90*. 1990.

6. Buxton, W., and B. Meyers. *A Study in Two-Handed Input*. in *Proc. of CHI '86: ACM Conference on Human Factors in Computing Systems*. 1986.
7. Cao, X., and R. Balakrishnan. *VisionWand: Interaction Techniques for Large Displays Using a Passive Wand Tracked in 3D*. in *ACM Symposium on User Interface Software and Technology*. 2003.
8. Cutler, R., and M. Turk. *View-based Interpretation of Real-time Optical Flow for Gesture Recognition*. in *IEEE Conference on Automatic Face and Gesture Recognition*. 1998.
9. Douglas, S., A. Kirkpatrick, and I. S. MacKenzie. *Testing Pointing Device Performance and User Assessment with the ISO 9241, Part 9 Standard*. in *Proc. CHI'99*. 1999.
10. Fitzmaurice, G.W., H. Ishii, and W. Buxton. *Bricks: Laying the Foundations for Graspable User Interfaces*. in *Proceedings of CHI 1995*. 1995.
11. Hinckley, K., and M. Sinclair. *Touch-Sensing Input Devices*. in *ACM CHI 1999 Conference on Human Factors in Computing Systems*. 1999.
12. Hinckley, K., R. Pausch, D. Proffitt, and N. Kassell. *Interaction and Modeling Techniques for Desktop Two-Handed Input*. in *ACM UIST 1998 Symposium on User Interface Software & Technology*. 1998.
13. Horn, B.K.P., *Closed Form Solution of Absolute Orientation Using Unit Quaternions.* Journal of the Optical Society, 1987. **4**(4): p. 629-642.
14. Jellinek, H.D., and S. K. Card. *Powermice and User Performance*. in *SIGCHI Conference on Human Factors in Computing Systems*. 1990.
15. Kjeldsen, R., and J. Kender. *Interaction with On-Screen Objects Using Visual Gesture Recogntion*. in *CVPR '97*. 1997.
16. Krueger, M., *Artificial Reality II*. 1991: Addison-Wesley.
17. Kurtenbach, G., and W. Buxton. *The Limits of Expert Performance Using Hierarchic Marking Menus*. in *Proceedings of InterCHI '93*. 1993.
18. Lenman, S., L. Bretzner, and B. Thuresson. *Using Marking Menus to Develop Command Sets for Computer Vision Based Hand Gesture Interfaces*. in *Proceedings of the Second Nordic Conference On Human-Computer Interaction*. 2002.
19. Letessier, J., and F. Berard. *Visual Tracking of Bare Fingers for Interactive Surfaces*. in *ACM Symposium on User Interface Software and Technology*. 2004.
20. MacKenize, I.S., *Fitts' Law as Research and Design Tool in Human-Computer Interaction*, in *Human-Computer Interaction 1992*. 1992. p. 91-139.
21. Quek, F., T. Mysliwiec and M. Zhao. *FingerMouse: A Freehand Computer Pointing Interface*. in *Proc. of Int'l Conf. on Automatic Face and Gesture Recognition*. 1995.
22. Rekimoto, J. *ThumbSense: Automatic Mode Sensing for Touchpad-based Interactions*. in *CHI 2003 Late Breaking Results*. 2003.
23. Rekimoto, J., and Y. Ayatsuka. *CyberCode: Designing Augmented Reality Environments with Visual Tags*. in *Designing Augmented Reality Environments (DARE 2000)*. 2000.
24. Turk, M., and G. Robertson, *Perceptual User Interfaces.* Communications of the ACM, 2000.
25. Wellner, P., *Interacting with Paper on the DigitalDesk.* Communications of the ACM, 1993. **36**(7): p. 86-97.
26. Wilson, A. *TouchLight: An Imaging Touch Screen and Display for Gesture-Based Interaction*. in *International Conference on Multimodal Interfaces*. 2004.
27. Wilson, A., and N. Oliver. *GWindows: Towards Robust Perception-Based UI*. in *First IEEE Workshop on Computer Vision and Pattern Recognition for Human Computer Interaction*. 2003.
28. Wu, M., and R. Balakrishnan. *Multi-finger and Whole Hand Gestural Interaction Techniques for Multi-User Tabletop Displays*. in *ACM Symposium on User Interface Software and Technology*. 2003.

Appendix

We present a technique where a flow field may be characterized as simultaneous rotation in the image plane, uniform scaling, and two-dimensional translation. If the hand is mostly rigid, this technique can be used to determine change in the hand orientation in the image plane, change in height above the keyboard, and so on.

For the flow field described by $\mathbf{x}_i = [x_i \quad y_i]^T$ and $\mathbf{dx}_i = [dx_i \quad dy_i]^T$, each point $\mathbf{x}_i$ moves to $\mathbf{x}'_i = [x'_i \quad y'_i]^T = \mathbf{x}_i + \mathbf{dx}_i$ by rotation θ in the image plane, uniform scaling s and translation $\mathbf{t}$:

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{4}$$

$$\mathbf{x}'_i = s\mathbf{R}\mathbf{x}_i + \mathbf{t} \tag{5}$$

We first solve for rotation. With means $\bar{\mathbf{x}} = \frac{1}{N}\sum_i \mathbf{x}_i$ and $\bar{\mathbf{x}}' = \frac{1}{N}\sum_i \mathbf{x}'_i$ we may solve for θ [13]:

$$\theta = \arctan\left[\frac{(x_i - \bar{x})(y'_i - \bar{y}') - (y_i - \bar{y})(x'_i - \bar{x}')}{(x_i - \bar{x})(x'_i - \bar{x}') + (y_i - \bar{y})(y'_i - \bar{y}')}\right] \tag{6}$$

Scaling factor s and translation $\mathbf{t} = [t_x \quad t_y]^T$ may be recovered by least squares:

$$\mathbf{z} = [s \quad t_x \quad t_y]^T \tag{7}$$

$$\mathbf{M}_i = \begin{bmatrix} \mathbf{R}\mathbf{x}_i & \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \end{bmatrix} \tag{8}$$

$$\mathbf{x}'_i = \mathbf{M}_i\mathbf{z} \tag{9}$$

$$\mathbf{z} = \left(\sum_i \mathbf{x}'^T_i \mathbf{M}_i\right)\left(\sum_i \mathbf{M}_i{}^T \mathbf{M}_i\right)^{-1} \tag{10}$$

It may not be obvious that this formulation allows for rotation and scaling about any point. For example, consider rotation about a point $\mathbf{t}_R$:

$$\begin{aligned} \mathbf{x}'_i &= s(\mathbf{R}(\mathbf{x}_i - \mathbf{t}_R) + \mathbf{t}_R) + \mathbf{t} \\ &= s\mathbf{R}\mathbf{x}_i + \mathbf{t}' \end{aligned} \tag{11}$$

where with $\mathbf{t}' = -s\mathbf{R}\mathbf{t}_R + s\mathbf{t}_R + \mathbf{t}$ we arrive at the original form of equation 5.