Multimodal Sensing for Explicit and Implicit Interaction

Andrew Wilson and Nuria Oliver

Microsoft Research
Redmond, WA
awilson@microsoft.com, nuria@microsoft.com

Abstract

We present four perceptual user interface systems that explore a continuum from explicit to implicit interaction. Explicit interactions include most of today's mouse and keyboard-based interaction models, where the user initiates a discrete action and expects a timely discrete response. Implicit interactions may use passive monitoring of the user over longer periods of time, and result in changing some aspect of the rest of the interaction. For example, less urgent notifications may be withheld from the user if the system detects they are engaged in a meeting. The first system is FlowMouse, a program that tries to emulate the mouse but suggests more implicit kinds of interaction. Second, we describe GWindows, which focuses complementing the mouse in today's GUI in a way that might support casual interactions. Then, ToughtLight eschews the traditional notion of an explicit discrete pointer altogether and in so doing presents a number of challenges in designing applications. And finally, S-SEER supports a purely implicit style of interaction driven by models of situational awareness. In presenting this series of projects in order from explicit to implicit modes, we hope to illustrate by way of example the various challenges and opportunities for perceptual user interfaces.

1 Introduction

Perceptual user interfaces [12] use cameras, microphones and other sensors to sense a computer user's motion, speech, activity and so on, for the purposes of interaction. Often the goal is to develop a mode of interaction that is considered more natural or fluid than that of today's desktop computing model, or to enable interactions in physical configurations in which the usual desktop interactions are inappropriate or unavailable.

Researchers applying perceptual user interfaces are faced with the fact that today's computing environments have grown up around an *explicit* style of interaction in which discrete actions are issued by the user, who in turn expects discrete responses from the system. While it is desirable to apply perceptual user interfaces to existing computing environments in order to speed adoption, it is often the case that they are not well suited to emulating the crisp input of the mouse and keyboard. For example, most of today's video cameras nominally capture images at a resolution of 640x480 pixels at 30 Hertz, while a commodity optical mouse possesses orders of magnitude more resolution and speed. However, perceptual UIs may have advantages even when emulating the mouse and keyboard. Projection keyboards, for example, promises ultra portable keyboarding on any flat surface [11].

A further problem in emulating traditional explicit interaction with perceptual UIs is the uncertain nature of the outputs of such systems. Often they are based on recognition results that do not work correctly every time, and failures can lead to breakdowns in the interaction that are costly to fix. Speech recognition errors, for example, are sometimes so difficult to recover from that they often negate the utility of using speech altogether.

This mismatch between the performance of perceptual UIs and the expectations built into the rest of our systems suggests that we are applying these novel techniques inappropriately, and/or should consider redesigning the system entirely; i.e., design a shell in which the perceptual UI is motivated and works well. At the very least, we should consider the value of perceptual UIs that do not emulate the mouse or keyboard directly, but rather complement these traditional devices instead. As a further step, we can consider supporting other kinds of interaction altogether different than pointing and text entry. One approach is to design an *implicit* style of interaction. These are interactions that neither require the discrete inputs of explicit interactions, nor give the user a expectation for a discrete timely response. What useful tasks can such a mode of interaction support?

In this paper we present four perceptual UIs that vary in terms of the explicit or implicit nature of their interaction. FlowMouse first tries to emulate the mouse but suggests more implicit kinds of interaction. GWindows is focused complementing the mouse in today's GUI in a way that might support casual interactions. ToughtLight eschews the traditional notion of an explicit discrete pointer altogether and in so doing presents a number of challenges in designing applications. S-SEER supports a purely implicit style of interaction driven by models of situational awareness. In presenting this series of projects in order from explicit to implicit modes, we hope to illustrate by way of example the various challenges and opportunities for perceptual user interfaces.

2 FlowMouse

Cameras and other sensors present opportunities to enrich the user experience. For example, computer vision techniques may be used to capture many details of human shape and motion. Considering the integration of cameras and vision techniques in today's computing experience, should such devices replace the mouse, complement the mouse, or lead to entirely new styles of interaction? With FlowMouse, we consider how vision techniques can be used with today's point and click interfaces, but with a view towards applying related techniques to novel interactions which go beyond emulating the mouse.

To capture the motion of the hand, FlowMouse uses optical flow techniques rather than traditional absolute position-based tracking methods. 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, makes few assumptions about the appearance of the hand, illumination levels, and so on..

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 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 prototype, a touch sensor is affixed to the surface of the left mouse button; this touch sensor is used to turn FlowMouse on and off in a natural way, as well as perform the "clutching" operation that is necessary in a relative motion framework.

While the user touches the left mouse button, FlowMouse sensing is enabled. During this time, grayscale images are

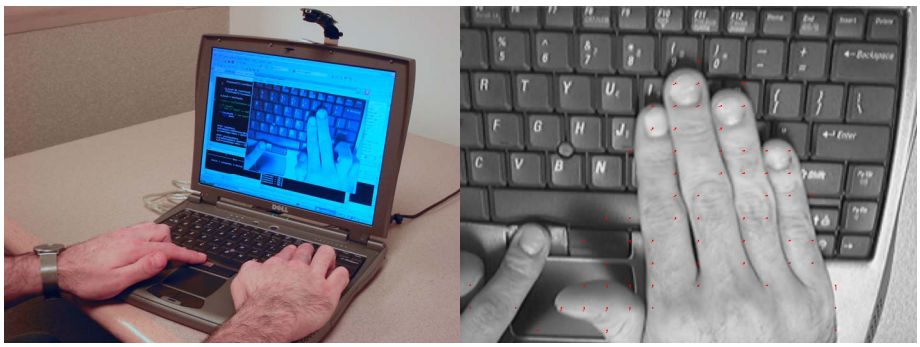

Figure 1 Left: FlowMouse prototype with screen-mounted camera facing down on keyboard and user's hands, and touch sensitive strip on left mouse button. Right: Example image capture from camera.

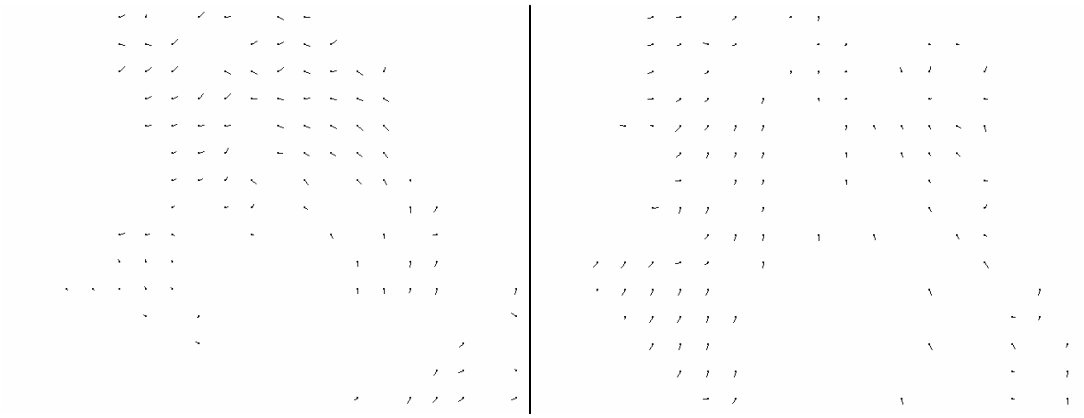

Figure 2 Flow fields represent more than translation. Left: Flow field generated by clockwise rotation of the hand. Note how the flow field vectors lie along circles about the point of rotation. Right: Flow field generated by hand raising over keyboard. Here the flow vectors indicate an expansion about a point.

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, for some regular grid defined on the input image, indicates the direction and magnitude of motion at each point on the grid [1], [2].

To use FlowMouse to drive the cursor, it is sufficient to simply average the flow field vectors together to obtain a relatively stable estimate of the hand's translation. When used with a mouse acceleration profile, FlowMouse is a usable replacement for the mouse. But flow fields are able to express patterns of motion beyond translation alone and have been shown to be useful in gesture recognition tasks [3]. Figure 2 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: For example, 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. 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. 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.

Other examples of applications beyond pointing include recovering tilting out of the image plane, supporting modeless scrolling or panning, simple gesture recognition such as "left" and "right" for "previous" and "next", and two handed interactions.

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.

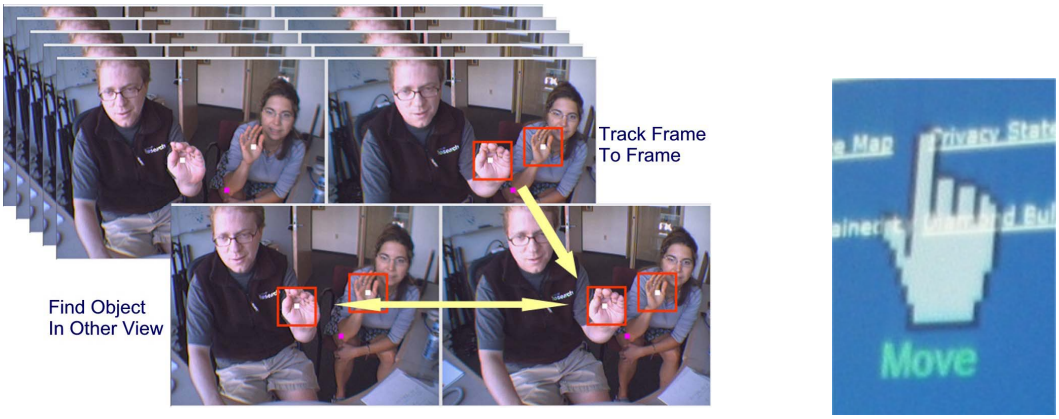

Figure 3 GWindows. Left: Illustration of stereo image processing to find moving, hand-sized objects in the scene. Only the nearest such object is labelled as the hand. Right: On-screen representation of hand position.

3 GWindows

The next level on the explicit-implicit axis is represented by GWindows, a real-time audio and stereo vision system that allows users to manipulate on-screen objects with gestures and voice [15]. As computers become more integrated in our daily lives, we can expect to find users in a variety of contexts where traditional mouse and keyboard interfaces are awkward, too intrusive or unavailable. Perceptual user interfaces have the potential to fill new roles in user experiences opened by these new scenarios, but there is presently an unfulfilled need for lightweight, robust and responsive sensing algorithms. The design of GWindows was guided by these constraints, including robust performance outside of the laboratory, low computational cost, use of common hardware, easy set-up and calibration, and non-intrusiveness.

GWindows uses video and audio as perceptual inputs. On the video side, we have developed a lightweight sparse stereo vision algorithm from images captured by two inexpensive off-the-shelf cameras (see Figure 3). On the audio side, we use Microsoft SAPI 5.1, with a simple command and control grammar and an inexpensive open microphone placed in front and below the display. The current system takes less than 20% of the CPU time on a 1GHz Pentium III.

The GWindows interface extends the usual WIMP interface enabling users to "grab" a window with their hands and move it across their desktop, close, minimize, maximize windows as well as scroll the foreground window.

Users explicitly initiate an interaction with GWindows by moving their hand across a predefined "engagement plane", an invisible plane about twenty inches in front of the display, and parallel to the plane of the display. When the hand crosses the engagement plane, feedback is given to the user by drawing a large alpha-blended hand icon on the usual Windows desktop. This icon is distinct from the usual Windows cursor and can be viewed as an area cursor [16]. The engagement plane is placed such that the user's hands do not enter it during the normal use of the mouse and keyboard. When the system is "engaged", the hand icon is moved to reflect the position of the user's hand. This is illustrated in Figure 3.

An open microphone for speech recognition is placed in front of and below the display. The user may invoke one of a small set of verbal commands in order to act upon the current window under the hand icon. When an utterance is understood by the system, the token phrase is drawn along with the icon to give feedback that the speech recognition system understood the utterance.

In addition to using speech to initiate modes of interaction such as moving or raising windows, the user may additionally operate GWindows by using hand gestures. In this case, the user may select the interaction mode by pausing or dwelling the hand over the target window. By dwelling a short amount of time (about .5 seconds), the target window is raised if it was not already the topmost window. If the hand dwells a longer amount of time (about

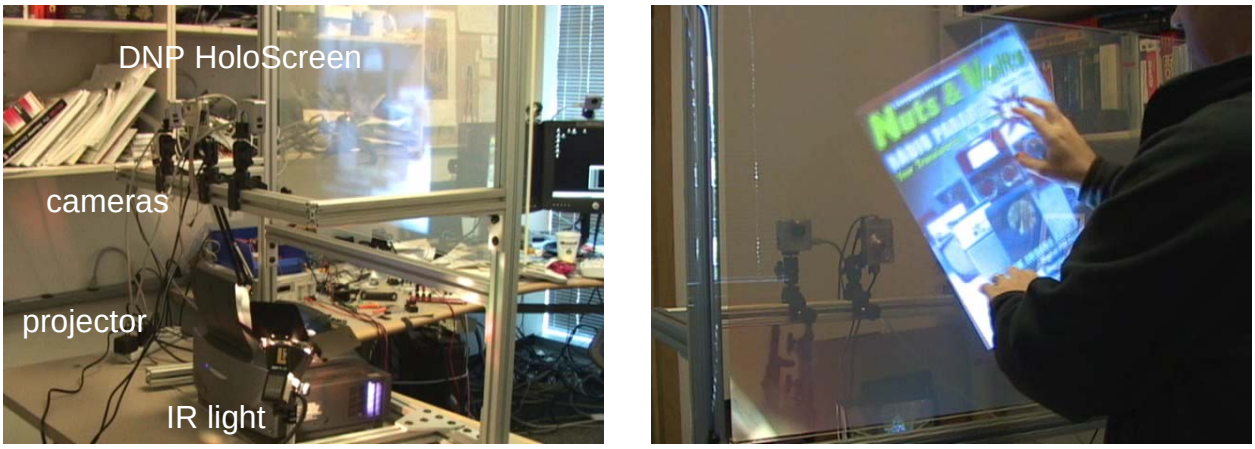

Figure 4 TouchLight prototype. Left: Placement of video cameras directly behind HoloScreen, short-throw projector, and infrared light. Right: High resolution still capture of magazine cover place on the display surface is directly manipulated by hand motion.

1 to 1.5 seconds), the hand icon text changes to "gesture", and the user may move the hand quickly left or right (a flick gesture) to send a window to the adjacent (left or right) monitor in a multi-monitor system. The system smoothly animates the movement of the window with a "swish" sound. If there is no adjacent monitor, the window is minimized. Finally, if the user dwells even longer (about 2 seconds), the mode changes to "Move" mode, where users can move the window by moving their hand. This change of interaction mode by dwelling relies on the continuous feedback of the mode label under the icon: a user simply pauses and dwells long enough until the desired mode is displayed. When the user then moves the hand, the system effects the mode's associated action (e.g. moving windows) and also exits the selection of modes.

We performed a preliminary, qualitative user study to determine how everyday users of GUIs find using GWindows. In this study, we confirm Kjeldsen's observations on a related system in which he found users become adept at selecting and moving items on the display very quickly, while new users tend to tire easily holding up their hand [6]. Unlike Kjeldsen's system, however, we rely on a small set of speech commands rather than requiring the user to put their hands in specific configurations to change application function. In our experiments, users were able to pick up the system very quickly, and many were pleasantly surprised to find how responsive the system is.

4 TouchLight

A number of recent research projects have explored post-WIMP (windows, icons, menus, pointer) interfaces situated on surfaces such as tables and whiteboards. These systems typically enable natural gesture-based input. For example, 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 [4], [17]. Many of these systems behave somewhat similarly to touch screens, but often enable multiple touch outputs and rich shape information. We call such systems imaging touch screens, to distinguish them from the usual touch screens.

Imaging touch screens are interesting for the present discussion in that while they offer capabilities that go beyond point and click interfaces, they are typically not well suited to point and click interfaces. For example, it is often difficult to determine which part of the finger should correspond with the precise two-dimensional cursor position. On the other hand, it is quite easy and natural to select an object on the surface by touching and manipulating it in more of a direct manipulation framework.

TouchLight uses simple computer vision techniques to compute a touch image on a plane situated between a pair of cameras and the user [14]. We demonstrate these techniques in combination with a projection display material which

Figure 5 Example of TouchLight image processing. Edge images from both cameras (left and middle) are combined to produce a touch image (right). In this case the hand on the left is flat on the display surface, while only the fingertips of the hand on the right are touching the surface.

permits the projection of an image onto a transparent sheet of acrylic plastic, and the simultaneous operation of the computer vision processes (see Figure 4).

TouchLight goes beyond the previous camera-based systems [8] [7] [13]; by not using a diffusing projection surface, it permits a high resolution touch image. For example, a high resolution image of a paper document may be captured using a high-resolution still camera, or one of the newer high resolution CMOS video cameras.The absence of a diffuser also permits the cameras to see beyond the display surface, just as they would if placed behind a sheet of glass. This allows a variety of interesting capabilities such as using face recognition techniques to identify the current user, eye-to-eye video conferencing, and other processes which are typically the domain of vision-based perceptual user interfaces.

The physical configuration of TouchLight is illustrated in Figure 4. A pair of commonly available Firewire web cameras are mounted behind the display surface such that each camera can see all four corners of the display. The DNP HoloScreen material is applied to the rear surface of the acrylic display surface. The HoloScreen is a special refractive holographic film which scatters light from a rear projector when the incident light is at a particular angle. The material is transparent to all other light, and so is suitable for applications where traditional projection display surfaces would be overwhelmed by ambient light.

The goal of TouchLight image processing is to compute an image of the objects touching the surface of the display, such as the user's hand. Due to the transparency of the display, each camera view shows the objects on the display and objects beyond the surface of the display, including the background and the rest of the user. With two cameras, the system can determine if a given object is on the display surface or above it. TouchLight image processing acts as a filter to remove objects not on the display surface, producing a *touch image* which shows objects that are on the display surface and is blank everywhere else. An example of one style of this image processing step is illustrated in Figure 5.The image processing steps are detailed in [14].

Many traditional computer vision algorithms may be used to derive features relevant to an application. For example, it is straightforward to determine the centroid and moments of multiple objects on the surface, such as hands. One approach is to binarize the touch image, and compute connected components to find distinct objects on the surface (see [5]). Such techniques may also be used to find the moments of object shapes, from which dominant orientation may be determined. Further analysis such as contour analysis for the recognition of specific shapes and barcode processing are possible.

We have also explored the application of optical flow, as alluded to in section 2, to the manipulation of onscreen objects. In this mode, mathematical techniques are used to choose a rotation, translation and scaling that best summaries the optical flow corresponding to motion on the screen surface. This composite transform is then applied directly to obtain a style of gesture-based interaction which is natural, fluid, and modeless.

5 S-SEER

Moving along the explicit-implicit interaction axis, the last system that we describe on this paper is S-SEER [9, 10], a real-time, multimodal office activity recognition system with selective perception.

A key challenge in inferring human-centric notions of context from multiple sensors is the fusion of low-level streams of raw sensor data –for example, acoustic and visual cues—into higher level assessments of activity. In S-SEER we developed a probabilistic representation based on a tiered formulation of dynamic graphical models named Layered Hidden Markov Models (LHMMs) [10]. We converged on the use of a multilayer representation that reasons in parallel at multiple temporal granularities, by capturing different levels of temporal detail. In LHMMs, each layer is connected to the next layer via its inferential results. The representation segments the problem into distinct layers that operate at different temporal granularities[1] --allowing for temporal abstractions from pointwise observations at particular times into explanations over varying temporal intervals. LHMMs can be regarded as a cascade of HMMs.

5.1 Architecture

Figure 6 illustrates S-SEER's three layer architecture:

1. At the lowest level we have the video and audio streams:
    a. On the audio side, S-SEER uses a pair of mini-microphones sampled at 44100KHz. Linear Predictive Coefficients are computed and feature selection is applied on these coefficients by means of principal component analysis. The number of features is selected such that at least 95% of the variability of the data is maintained, which is typically achieved with 7 features. The energy, the mean and variance of the fundamental frequency and the zero crossing rate are also computed. The source of the sound is localized using the Time Delay of Arrival method.
    b. On the video side, S-SEER uses a standard Firewire camera sampled at 30 frames per second. Four features are extracted from the video signal: the number of skin color, motion, foreground and face pixels in the image.
    c. Finally, a history of the last 1, 5 and 60 seconds of mouse and keyboard activities are logged.

2. **First Level HMMs**: The first level of HMMs includes two banks of distinct HMMs for classifying the audio and video feature vectors. With respect to the audio signal, there is one HMM for each of the following typical office sounds: *human speech, music, silence, ambient noise, phone ringing* and the sounds of *keyboard typing*. All the HMMs are run in parallel. At each instant, the HMM with highest likelihood is selected and the data is classified correspondingly. The video signals are classified using another bank of HMMs to detect whether there *is nobody, one person present, one active person present* or *multiple people present* in the office.

3. **Second Level HMMs**: The inferential results from the previous layer, i.e. the outputs of the audio and video classifiers, the derivative of the sound localization component, and the history of keyboard and mouse activities constitute a feature vector that is passed to the highest level of analysis. This layer handles concepts with longer temporal extent, i.e. the user's typical activities in or near an office. In particular, the activities modeled are: *phone conversation, presentation, face-to-face conversation, user present,* engaged in some other activity*; distant conversation* (outside the field of view); and *nobody present.* Some of these activities can be used in a variety of ways in services, such as those that idenfity a person's availability.

[1] The concept of "temporal granularity" in this context corresponds to the window size or vector length of the observation sequences in the HMMs.

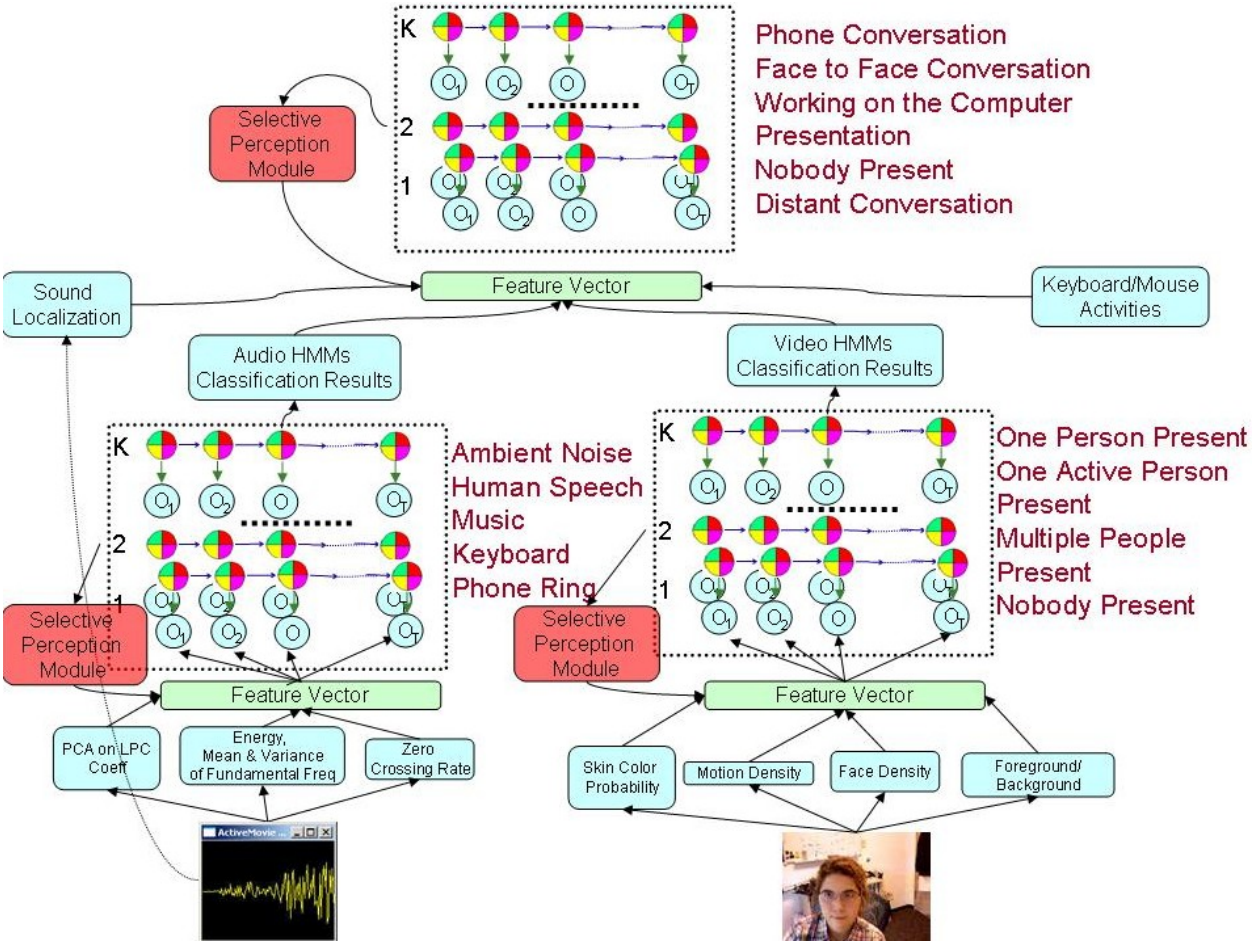

Figure 6 S-SEER architecture

5.2 Selective Perception

Most perceptual interfaces consume a large portion of the available CPU, due to having to process computationally very intense sensor information (e.g. video, audio) to make inferences about the situation. To address this issue, we integrated in S-SEER a principled, decision-theoretic approach for guiding perception. We worked to apply the *expected value of information (EVI)* to determine dynamically which features or sensors to use, depending on the context.

We compute the expected value of information for a perceptual system by considering the value of eliminating uncertainty about the state of the set of features or sensors under consideration. We refer the reader to [10] for a more detailed description of our selective perception algorithms.

5.3 Experimental Results

We refer the reader to [9, 10] for more detailed descriptions of our experimental results. In an initial set of experiments, we compared the recognition accuracy of LHMMs when compared to HMMs. We trained and tested LHMMs and HMMs on 60 minutes of recorded office activity data (10 minutes per activity, 6 activities and 3 users). We used 50% of the data for training and the rest for testing. The average accuracy of LHMMs was **99.7%** (STD 0.95), much higher than that of HMMs, **72.7%** (STD 8.15). In our experience, HMMs normally need training under

similar office conditions (lighting, acoustics, etc) than that of the particular testing data to obtain reasonable classification results. LHHMs are much more robust to environmental changes.

Next, we considered the diagnostic accuracy and the computational cost incurred by the system when using selective perception. The results are displayed in Tables 1 and 2. We use the abbreviations: PC=Phone Conversation; FFC=Face to Face Conversation; P=Presentation; O=Other Activity; NP=Nobody Present; DC=Distant Conversation.

Tables 1 and 2 compare the average recognition accuracy and average computational cost (measured as % of CPU usage) when testing S-SEER on 600 sequences of office activity (100 sequences/activity) with and without selective perception. Note how S-SEER with selective perception achieved as high level of accuracy as when using all the sensors all the time, but with a significant reduction of CPU usage.

	PC	FFC	P	O	NP	DC
No Selective Perception	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
EVI-based Selective Perception	100.0%	100.0%	97.8%	100.0%	98.9%	100.0%

Table 1. Recognition accuracies of LHMMs without and with selective perception.

	PC	FFC	P	O	NP	DC
No Selective Perception	61.2%	67.1%	50.0%	59.0%	44.3%	44.5%
EVI-based Selective Perception	44.5%	56.5%	21.0%	19.6%	35.7%	23.3%

Table 2. Average computational costs of S-SEER without and with selective perception.

Some interesting observations that can be noted from our experiments are: (1) At times, the system chooses not to use any features at all, as S-SEER is confident enough about the situation, and it selectively turns the sensors on only when necessary; (2) in most experiments, S-SEER never turned the sound localization feature on, due to its high computational cost versus relatively low informational value this acoustical feature provides.

6 Conclusion

We presented four perceptual user interface systems which vary widely in terms of the nature of the interaction enabled by adding cameras and other sensors to reason about and react to the user in intelligent ways. FlowMouse for example begins by emulating the very explicit command-driven style of the point and click interface, while at the other end of the spectrum, S-SEER enables office awareness applications driven by ongoing patterns of activity and not explicit actions on the part of the user.

By considering the degree of implicit or explicit interaction enabled by perceptual user interfaces, we may draw attention to the many ways that such novel sensing systems may be applied and how they may influence the design of tomorrow's systems. While the application of these techniques to today's computing systems may be attractive in terms of ease of adoption, consider that today's graphical user interfaces (GUIs) have been designed around the capabilities of the mouse, and a very narrow notion of computer-human interaction based on pointing and clicking. The analogous re-conception of the interface around perceptual user interfaces has not yet happened.

However, as computers become ubiquitous, the variety of computing form factors continues to increase, and as these devices become more a part of our daily lives, there will be demand for modes of interaction that go far beyond existing interfaces. Consider the explosive growth in the variety of form factors: mobile devices, living room PCs, tablet PCs, game consoles, auto PCs, and so on. Each of these demands new thinking around the interface. We believe that a number of these systems will operate on more implicit levels of interaction than we see today, while the ever present explicit interfaces will offer richer, more fluid and more contextually appropriate modes of interaction. Both styles of interaction can benefit from the unique capabilities of perceptual user interfaces.

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. 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.
4. Fitzmaurice, G.W., H. Ishii, and W. Buxton. *Bricks: Laying the Foundations for Graspable User Interfaces*. in *Proceedings of CHI 1995*. 1995.
5. Horn, B.K.P., *Robot Vision*. 1986, Cambridge, MA: MIT Press.
6. Kjeldsen, R., and J. Kender. *Interaction with On-Screen Objects Using Visual Gesture Recogntion.* in *CVPR '97*. 1997.
7. Klemmer, S.R., M. W. Newman, R. Farrell, M. Bilezikjian, J. A. Landay. *The Designer's Output: A Tangible Interface for Collaborative Web Site Design*. in *ACM Syposium on User Interface Software and Technology*. 2001.
8. Matsushita, N., J. Rekimoto. *HoloWall: Designing a Finger, Hand, Body and Object Sensitive Wall*. in *ACM Symposium on User Interface Software and Technology (UIST)*. 1997.
9. Oliver, N., and E. Horvitz. *Selective Perception Policies for Guiding Sensing and Computation in Multimodal Systems: A Comparative Analysis*. in *Proc. Int. Conf. on Multimodal Interfaces*. 2003.
10. Oliver, N., E. Horvitz, and A. Garg, *Layered Representations for Human Activity Recognition.* Computer Vision and Image Understanding, 2004. **96**(2): p. 163-180.
11. Tomasi, C., A. Rafii, and I. Torunoglu, *Full-size Projection Keyboard for Handheld Devices.* Communications of the ACM, 2003. **46**(7): p. 70-75.
12. Turk, M., and G. Robertson, *Perceptual User Interfaces.* Communications of the ACM, 2000.
13. Ullmer, B., H. Ishii. *The metaDESK: Models and Prototypes for Tangible User Interfaces*. in *ACM Symposium on User Interface Software and Technology*. 1997.
14. Wilson, A. *TouchLight: An Imaging Touch Screen and Display for Gesture-Based Interaction*. in *International Conference on Multimodal Interfaces.* 2004.
15. 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.
16. Worden, A., N. Walker, K. Bharat, and S. Hudson. *Making Computers Easier for Older Adults to Use: Area Cursors and Sticky Icons.* in *ACM Conf. on Human Factors in Computing Systems, CHI '97*. 1997.
17. 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.