Hoverflow: Exploring Around-Device Interaction with IR Distance Sensors

Sven Kratz
Deutsche Telekom Laboratories
TU Berlin, Germany
+49 30 8353 58289
sven.kratz@telekom.de

Michael Rohs
Deutsche Telekom Laboratories
TU Berlin, Germany
+49 30 8353 58469
michael.rohs@telekom.de

ABSTRACT
By equipping a mobile device with distance sensing capabilities, we aim to expand the interaction possibilities of mobile and wearable devices beyond the confines of the physical device itself to include the space immediately around it. Our prototype, an Apple iPhone equipped with six IR distance sensors, allows for rich 3D input, comprising coarse movement-based hand gestures, as well as static position-based gestures. A demonstration application, HoverFlow, illustrates the use of coarse hand gestures for interaction with mobile applications. This type of interaction, which we call Around-Device Interaction (ADI) has the potential to help to solve occlusion problems on small-screen mobile devices and scales well to small device sizes.

Categories and Subject Descriptors
D.3.3 [**Programming Languages**]: Language Contructs and Features – *abstract data types, polymorphism, control structures.* This is just an example, please use the correct category and subject descriptors for your submission. The ACM Computing Classification Scheme: http://www.acm.org/class/1998/

General Terms
Your general terms must be any of the following 16 designated terms: Algorithms, Management, Measurement, Documentation, Performance, Design, Economics, Reliability, Experimentation, Security, Human Factors, Standardization, Languages, Theory, Legal Aspects, Verification.

Keywords
Keywords are your own designated keywords.

1. INTRODUCTION
Using sensors, the interaction space of small mobile devices can be extended beyond the physical boundary of mobile devices to include the full 3D space around them. We call the resulting interaction space Around-Device Interaction (ADI). Around-device interaction has the potential to be a beneficial addition to standard interface elements of mobile devices, such as keypads or touch screens. This is particularly attractive for very small devices, such as wristwatches, wireless headsets, and future types of wearable devices such as digital jewelry (Figure 1).

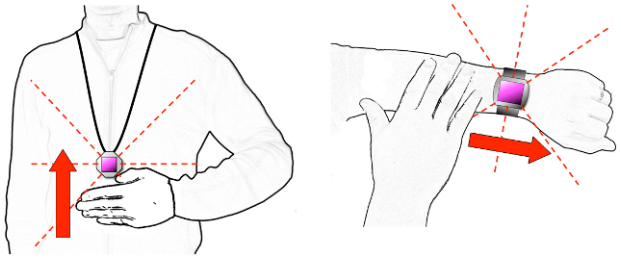

Figure 1. Interacting with very small devices via coarse gestures. The gestures are detected by an array of proximity sensors extending in radial direction from the device.

The space beyond the device, however, can easily be used, no matter how small the device may be. Such wearable devices can also serve as easily accessible controllers for appliances in the environment or for wireless communication applications. In a smart home environment, for example, a gesture on the device could dim the light or control the volume of entertainment system.

In this demonstration we present a prototype ADI-based interface using an Apple iPhone augmented with six IR distance sensors. A demonstration application, HoverFlow, shows how coarse hand gestures, which are performed in the vicinity above the device, can be used to browse and select colors from a palette in a mobile application. In the following, we will give a detailed description of the hardware and software implementation of our prototype and show a number of movement-based gestures that are suitable for recognition by proximity sensors. Furthermore we will discuss why this prototype is a valuable exhibit to be shown as a demonstration at Mobile HCI 2009.

2. RELATED WORK
The Gesture Pendant, developed by Starner et. al. [], is a small mobile device that is worn around the neck as a pendant. Using a camera, it enables the detection of hand gestures to control a home automation system. In contrast to Gesture Pendant, our work uses a substantially cheaper tracking system. Additionally, the reliable distance measures provided by the IR sensors allow for simple software implementation of the gesture recognition.

MobileHCI'09, September 15 - 18, 2009, Bonn, Germany.
ACM 978-1-60558-281-8.

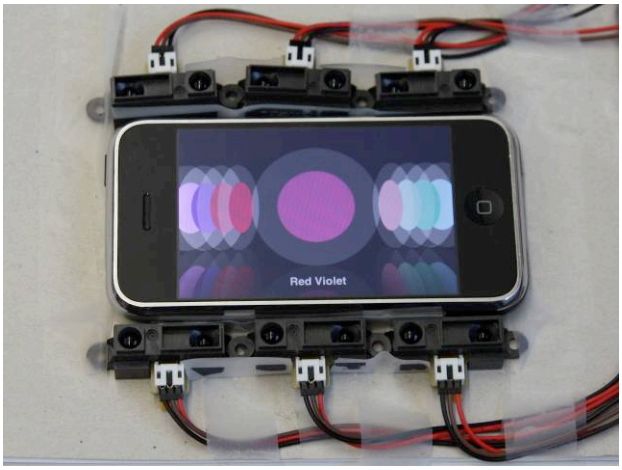

Figure 2. The current set-up of our prototype. Six Sharp GP2D120X IR distance sensors are placed along the long edges of an iPhone mobile device running the HoverFlow application (described in Section Fehler! Verweisquelle konnte nicht gefunden werden.**). Using simple hand gestures, the user can scroll and select colors in the color palette.**

Hinckley et. al. adopted the idea of placing an infrared (IR) distance sensor on a mobile device and investigated technical characteristics of this kind of sensor technology [5]. The infrared distance sensor allowed the device to detect the presence of the user. This idea is used today in a number of Digital Single Lens Reflex (DSLR) cameras that switch off the LCD display on the back when the user looks through the viewfinder.

SideSight [3] is an instance of an around-device interface by locating a series of IR sensors on the long edges of a small mobile device. This technique allows capturing simple multi-touch gestures around the device's perimeter. SideSight focuses on minimizing occlusion problems and is designed for operation while the device is placed on flat surfaces. In SideSight, the field of the distance sensors extends across the display surface to the left and right of the device. In our prototype the sensors are oriented towards the user to allow for handheld interaction (one hand is holding the device, the other hand performs the gesture). In more flexible setups, the distance sensors should be oriented in multiple directions to cover the whole space around the device.

Baudisch and Chu [2] focus on adding pointing input capabilities to very small devices. In order to avoid occlusion they use a touch screen on the back of the device and show that this approach is successful even for display sizes below 1". Since interaction with the nanoTouch device [2] means touching the back of the device, the possible physical extent of input movements is still given by the size of the device. In contrast, around-device interactions are independent of the physical size

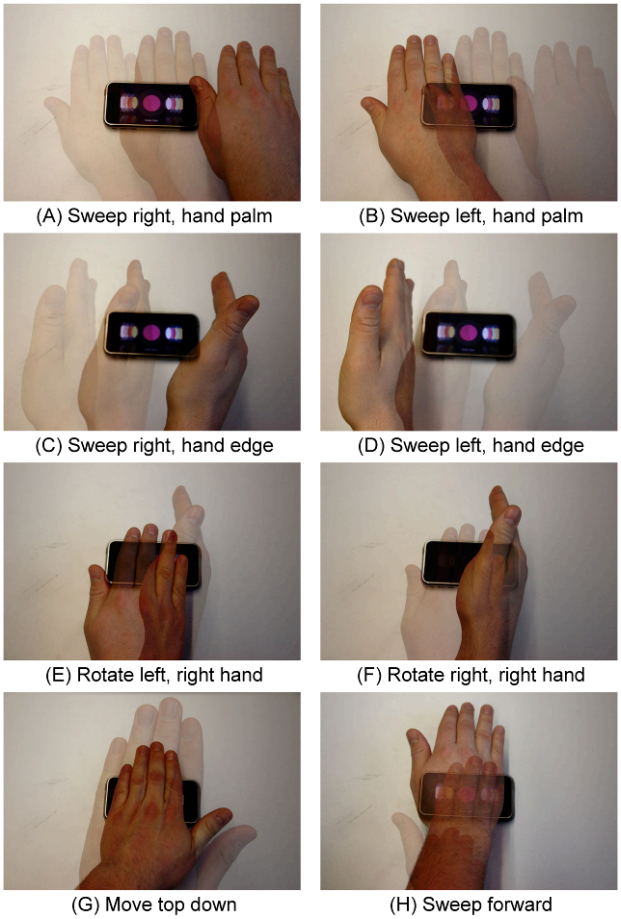

Figure 3. An overview of the hand gestures currently recognized by our prototype.

3. HOVERFLOW

Our *HoverFlow* is an example application for the Apple iPhone that demonstrates the use of a sensor-based interface for detecting coarse hand-gestures above small mobile devices. The implementation of our application is partially based upon the *CoverFlow Example* by Sadun [13]. HoverFlow allows the user to select colors from a color palette through hand. Possible gestures are moving the hand across the device, presenting a number of hand postures, or by moving a hand rapidly towards or away from the device. Figure 3 shows an overview of all gestures currently implemented in our system.

3.1 Supported Gestures

The *CoverFlow View* provided by the iPhone's iPod application inspired the visual layout of HoverFlow. Thus, we decided to map the user's movements in the following way: if her hand moves across the device from left to right (Figure 3 A), the color palette scrolls from left to right, and vice-versa (Figure 3 B). A hand-edge movement from left to right (Figure 3 C) makes the color palette scroll 5 colors to the right and vice-versa (Figure 3 D). A color is selected when the user moves her hand swiftly towards the device (Figure 3 G). A color is deselected when the user moves her hand rapidly away from the device. Rotating the hand towards the left (Figure 3 E) or right (Figure 3 F) permits the user

to scroll directly to the beginning or end of the palette, respectively.

3.2 Interface Implementation

3.2.1 Sensing

To capture simple hand movements and gestures, our prototype system uses six Sharp GP2D120X IR [16] distance sensors, placed around the device's edges and facing vertically away from the device. Figure 2 shows the current sensor configuration of our prototype.

An Arduino BT [1] microcontroller board captures the distance readings provided by the sensors. The sensors supply 256 discrete range readings allowing them to detect distant objects from 4 to 30 cm away. The sensor update rate is 25Hz. A PC processes the sensor data, and handles the gesture recognition. In future versions of HoverFlow, we aim to conduct all processing on the mobile device, by establishing a direct link between the Arduino board and the mobile device via RS-232 or Bluetooth.

3.2.2 Gesture Detection and Recognition

To smooth the raw sensor data, it is passed through a Savitzky-Golay filter [15] in an initial processing step. The filtered data is then added to a queue containing the differences of the last 16 sensor readings, i.e. the difference $\Delta D = D_t - D_{t-1}$. We use the difference values instead of the absolute values in order to make gesture recognition independent of the distance between the user's hands and the device. The queue is updated every time the Arduino provides a new sensor reading. The window length of 16 was chosen because the sampling rate of the distance sensors is 25 Hz, which means that the system constantly keeps a history of the last 640 ms of interaction. This window length provides us with enough samples do discern user gestures in a meaningful way while at the same time assuring a response time from the system within a acceptable time interval (<1000 ms).

An advantage of the method we implemented is that it does not require any clutching mechanism to detect the start and end of a gesture, which is often required for accelerometer-based gesture recognition. When no IR-reflective object is present in the range of the distance sensors, they will provide a noise floor of values close to zero. Gestures can be distinguished from operation on the touch screen, by checking whether the screen was touched after the distance sensors detect an object in range. If a screen touch event occurs then this activity is interpreted as touch input and the gesture is discarded. Otherwise the activity is treated as a gesture. Accelerometers constantly provide sensor data as the user moves. It is therefore much harder to distinguish between moves that are part of a gesture and those that are not.

To determine if a significant user movement has been detected, the Euclidean norm of the oldest element of the readings queue is constantly calculated. If this norm surpasses a predefined threshold, the remaining 15 sensor readings are analyzed to determine the end of the sequence representing user input. Interaction with HoverFlow is designed to take place within a certain distance range around the device, so this threshold is set to the value the sensor array provides when a large object is held in front of them at a distance of about 5-7 cm away from it.

3.2.3 Gesture Classification

Once the bounds of the sequence containing user activity have been detected, a best-matching gesture template from a set of prerecorded user inputs is estimated using Dynamic Time Warping (DTW). A good overview of how DTW functions can be found in [11, 14]. Gestures and templates are represented as 16-by-6 matrices of sensor value deltas.

DTW performs well in cases where the captured sample and the matching template are distorted in time, but have similar values. In our case, using DTW allows the recognition of gestures that are similar in movement to but are performed at different speeds than the pre-recorded templates. In our prototype, we achieved acceptable recognition rates using only 2 to 3 training samples per gesture, with a gesture vocabulary of up to 9 gestures.

DTW-based approaches generally need less training samples than other methods, such as Hidden Markov Models [21]. Thus we do not require an extensive corpus of gestures to be available in order for our prototype to function correctly. A possible drawback of the DTW algorithm, its time and space complexity of $O(n^2)$, is not an issue due to the small size of the sampling window, which results in a maximum size of the distortion matrix of 256 elements. (Entry (i,j) of the distortion matrix contains the DTW-distance between samples 1 to i of the gesture and samples 1 to j of the template. Entry (16,16) thus contains our measure of similarity between gesture and template. The distortion matrix is built up from entry (1,1) using a dynamic programming approach.) Because of the small size of the distortion matrix, CPU and memory requirements should not present a constraint for our algorithm if it is run on modern mobile devices. If, however, sensors with a higher sampling rate were to be employed, which would result in larger data sets to be processed at a time, it may be likely that further optimizations of the DTW algorithm, such as FastDTW [14], will be required.

3.2.4 Update of Mobile User Interface

Once a gesture has been detected, the user interface of the mobile device running the HoverFlow application needs to be updated. In our prototype, the PC sends XML-RPC calls to the mobile device to signal interface updates when new gestures have been detected.

4. ADVANTAGES OF AROUND-DEVICE INTERACTION

We built our prototype to demonstrate the advantages of Around-Device Interaction (ADI). ADI shows promising potential as an interface technology complimenting existing mobile device interfaces.

The occlusion problem on small device displays is at least partially solved by ADI, as implemented in HoverFlow. Because the user interacts with the device at a certain distance from its screen, a gap opens up between the user's hand and the display, allowing the user to see the display's contents at an angle.

Using ADI, fast but coarse interfaces can be implemented; in the case where the desired actions are so simple that fine-grained input using the device's keypad or touch-screen is not necessary.

Although ADI breaks the metaphor of direct manipulation [3], quick hand gestures may be particularly useful for tasks of an immediate and direct nature. Also, situations where visual interaction is not preferable, for example when driving vehicles, may benefit from interfaces that allow the input of simple commands using rough hand gestures.

5. ONGOING IMPROVEMENTS

By the time of demonstration we aim to have an even better prototype on our hands. We are currently working on these specific points

- further improvement of the gesture recognition algorithm
- add continuous interaction modes, i.e. gestures allowing the control of a continuous parameter, such as volume
- implementation of on-screen feedback to aid gesture input and to continuous modes of interaction
- implementation of further applications for the demonstrator, such as (but not limited to) a web-browser, media player, call-placement tool
- Upgrade of the sensor package to use a larger number of smaller, SMD, sensors to further improve the quality of the interface
- 3D printing of casing that combines the host device and the sensor package

6. SUMMARY

We demonstrate a prototype of a mobile user interface, which allows mobile devices to track coarse hand gestures using a small number of IR distance sensors. Our example application, Hoverflow, demonstrates the concept of expanding the interaction area of mobile devices beyond their physical boundaries. Our prototype shows how this technique can enhance mobile interaction either as a complimentary technique co-existing with existing interface technology or as a stand-alone interface for very small devices.

7. REFERENCES

[1] Arduino: prototyping boards and development environment, http://www.arduino.cc .

[2] Baudisch, P. and Chu, G. 2009. Back-of-device interaction allows creating very small touch devices. In Proc. of CHI ‘09.

[3] Butler, A., Izadi, S., and Hodges, S. 2008. SideSight: Multi-"touch" interaction around small devices. In Proc. of UIST '08. ACM, 201-204.

[4] Hinckley, K., Pierce, J., Sinclair, M., and Horvitz, E. 2000. Sensing techniques for mobile interaction. In Proc. of UIST '00. ACM, 91-100.

[5] Kruskall, J. & M. Liberman. The Symmetric Time Warping Problem: From Continuous to Discrete. In Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison, pp. 125-161, Addison-Wesley Publishing Co., Reading, Massachusetts, 1983

[6] Sadun, E., iPhone Developer’s Cookbook, Pearson Education, November 28, 2008, ISBN 0321555457.

[7] Salvador, S. and Chan, P., FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space, KDD Workshop on Mining Temporal and Sequential Data, 70-80, 2004

[8] Savitzky, A., Golay, M.J.E. 1964. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. Analytical Chemistry, 36 (8), 1627-1639.

[9] Starner, T. and Auxier, J. and Ashbrook, D. and Gandy, M. The gesture pendant: A self-illuminating, wearable, infrared computer vision system for home automation control and medical monitoring. International Symposium on Wearable Computing 2000. Pp. 87-94.