# HandSense - Discriminating Different Ways of Grasping and Holding a Tangible User Interface

**Raphael Wimmer**
University of Munich
Amalienstr. 17
80333 Munich, Germany
raphael.wimmer@ifi.lmu.de

**Sebastian Boring**
University of Munich
Amalienstr. 17
80333 Munich, Germany
sebastian.boring@ifi.lmu.de

## ABSTRACT
As mobile and tangible devices are getting smaller and smaller it is desirable to extend the interaction area to their whole surface area. The HandSense prototype employs capacitive sensors for detecting when it is touched or held against a body part. HandSense is also able to detect in which hand the device is held, and how. The general properties of our approach were confirmed by a user study. HandSense was able to correctly classify over 80 percent of all touches, discriminating six different ways of touching the device (hold left/right, pick up left/right, pick up at top/bottom). This information can be used to implement or enhance implicit and explicit interaction with mobile phones and other tangible user interfaces. For example, graphical user interfaces can be adjusted to the user's handedness.

### Author Keywords
touch, grasp, capacitive sensing, sensors, input devices, handedness

### ACM Classification Keywords
H5.2. Information interfaces and presentation (e.g., HCI): User Interfaces

## INTRODUCTION
As mobile and tangible devices are getting smaller and smaller it is desirable to extend the interaction area to their whole surface area. This can be done by making the surface touch sensitive. However, the shape of most devices does not facilitate complex interaction like 2D pointing - PDAs and tablet PCs being an exception. For small tangible user interfaces (TUIs) and mobile devices, the usable surface area often is required for holding the device, not allowing further interaction with it. However, the way of holding or grasping a device can also be used to - implicitly or explicitly - convey information. Taking into account the posture of the user's hand increases the expressiveness of interaction. The most obvious parameter is the pressure with which the object is held. The contact areas at which an object is touched and the amount of fingers used for grasping an object can also provide information. Another hint is whether the object is held between fingers or in the palm. Finally, the information which hand is holding the object - left or right - can be useful for explicit or implicit interaction. All of these parameters can be utilized in the design of touch-based user interfaces.

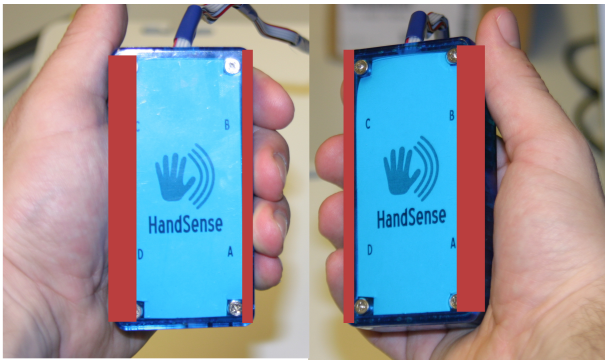


**Figure 1. HandSense is able to determine whether a device is held in the left or right hand by measuring the capacitance to each side. As the ball of the thumb is thicker than the fingers, it causes higher sensor readings.**

Our HandSense prototype (Figure 1) employs capacitive sensors for detecting when it is touched, squeezed, or held against a body part. HandSense is also able to detect in which hand the device is held, and how. As touching and squeezing has already been discussed in prior work [2, 6], we describe how capacitive sensors can be used for detecting a user's handedness and the way she is holding an object. We have verified the feasibility of our approach with a user study. Additionally, we provide examples of applications that can be enhanced by grasp and handedness detection.

## SENSING TOUCH AND PROXIMITY
Detecting touch and using it for interaction with mobile devices has been researched for quite some time. Most related work focuses on mobile devices with a display, like PDAs or mobile phones. Harrison et al. [2] were able to detect in which hand a user held a tablet computer by embedding pressure sensors in its back cover. When holding the device with her left hand the user triggered only sensors on the left


*TEI 2009*, February 16 - 18, 2009, Cambridge, UK.

edge and vice versa. This approach only works for devices that are held on different sides depending on which hand is used. For palm-sized devices pressure sensors can only detect squeezing, not handedness. Wigdor et al [5] double the interaction area of a PDA by making its back touch sensitive. The prototype uses a video camera behind the PDA to simulate this. When holding the PDA with both hands, all fingers except the thumbs are on the back of the PDA and can be used for manipulating screen content. This effectively reduces screen occlusion. Hinckley et al. [3] embedded an infrared proximity sensor, a capacitive touch sensor, and a tilt sensor into a PDA. These were used for several explicit interactions like scrolling text on the screen Mäntyjärvi et al. [4] measured impedance of the object touching a mobile phone using two conductive pads embedded into one side of the phone. They were able to distinguish a human hand from wet cotton fabric touching the sensors. Impedance measurements require electrical contact between sensor and touching object. Butler et al. [1] use an array of miniature infrared range finders embedded into the sides of a smartphone for tracking fingers in proximity. They mainly explore multi-touch gestures which require the phone to be put on a surface. However, such sensors could also detect grasp patterns.

Capacitive sensors are very sensitive to human tissue or other conductive objects. They are not particularily sensitive to furniture, plastics, or fabric. The sensors can be embedded inside casings as they do not need a line of sight or contact. Capacitive sensors are capable of sensing proximity up to a range of several inches, depending on antenna size. A major challenge when using capacitive sensors is the ambiguous data they provide. The same change in sensor readings can be effected by an object moving closer, by an additional object coming within sensor range, or by a shape change (like making a fist). A unique feature of capacitive sensors is that they can be used to estimate the thickness of an object in close proximity if distance, size, and electrical properties are known. However, controlling those parameters is difficult. But if two objects with identical electrical properties are at the same distance to a sensor and cover the same amount of the sensor area, the thickness of the objects can be compared. The thicker object will cause higher sensor readings than the thinner one. We utilize this effect for determining whether a capacitive sensor is covered by fingers or the ball of the thumb. As the ball of the thumb is distinctly thicker than the fingers, it causes higher sensor readings than these. Putting sensors on opposite sides of a palm-sized TUI allows for determining which side of the object faces the fingers, and which side faces the ball of the thumb. Putting the sensors on opposite sides also ensures that the pressure applied towards the sensors is the same for both sensors if the object is held in the palm. This is necessary to maintain the same distance between tissue and sensor. As the position of fingers and ball of the thumb are inverted between left and right hand, the TUI can thus infer, in which hand it is held.

## HANDSENSE PROTOTYPE

For our prototype implementation we used CapToolKit [6], an open-source capacitive sensing toolkit[1]. The sensors - based on 555 timer ICs - are connected to a CapBoard controller. The CapBoard controls the sensors and filters their readings. CapToolKit has two advantages over most commercial capacitive sensors: it outputs high-resolution proximity data, and its sensors employ a guard electrode for actively shielding the antenna. The HandSense prototype (Figure 1) is a plastics box about the size of a mobile phone (100 x 50 x 25 mm). It contains four capacitive sensors. We installed two sensors on each of the 100x25 mm sides. When holding the device in one's palm one pair of sensors faces thumb and wrist, the other pair faces the four fingers. Each sensor has an antenna measuring 30x15 mm, made of a tin sheet. Sensors are actively shielded by guard electrodes wrapping the antennas on three sides. This focuses the direction of the electric field, reducing noise and improving sensitivity. Due to space limitations we omitted sensors for the top and bottom sides, as well as for the small sides on the upper and lower end of the box. The four sensors are connected to the external CapBoard controller and sampled at 25 Hz. The CapBoard is connected to a host PC via USB. A Python program on the host PC analyzes and visualizes the sensor readings. It employs the CapToolKit Python API for communication with the CapBoard and PyGame for displaying the recognized state. We use the default settings for CapBoard, only disabling "stuck channel detection". This filter would reset sensor readings for a channel if they stayed the same for several seconds. As our participants should hold the prototype for some seconds, the filter might interfere with our detection algorithms. Additionally, we normalize all sensor readings to [0.0;1.0]. This is necessary because the sensors have different sensitivity and therefore different maximum readings. Calibration is done once when the system is turned on by covering all sensors tightly, effecting maximum sensor readings for each channel. Different levels of normalized sensor readings are mapped to different states as follows:
**No Proximity** (0.0 - 0.03) - sensor readings are very low. No body part is within sensor range.
**Near** (0.03 - 0.2) - sensor is near a body part but not touching it.
**On Hand** (0.2 - 0.5) - sensor is very close to a body part but not covered by it.
**Gripped** (0.5 - 0.85) - sensor is partially covered by a body part.
**Held** (0.85 - 1.0) - sensor is completely covered by body part.
Using these levels our prototype can distinguish 6 different grasping states (Figure 2). Additionally, it can detect whether the device is in a pocket or lies on a table. The levels and the following heuristics are based on initial assumptions and tuned by trial and error.

**On Table**. If all sensor readings are below 0.03 no hand is in proximity. It is assumed that the device is lying somewhere, for example on a table.
**In pocket**. If all sensor readings are within the "Near" range the device might be in a pocket.
**On hand**. If all readings are within the "On Hand" range,

[1] current version available at www.capsense.org

**Figure 2. HandSense can distinguish six different ways of grasping a device. From left to right: hold up, pull out, grasp right, grasp left, hold left, hold right.**

the device is probably lying on a hand or other body part.
**Grasped with left/right hand**. If all sensors but the sensor in the lower right side are within the "Gripped" range, the device is probably being grasped with the left hand. If only the sensor in the lower left is not within this range, the device is being grasped with the right hand.
**Grasped at top/bottom**. If only the readings of sensors at the upper or lower side of the device are in the "Gripped" range, the device is probably held with two fingers at the top or bottom end. Sensor readings do not allow for reliable discrimination of left and right hand.
**Held in left/right hand**. If all sensors readings are in the "Held" range, the device is probably held tightly in a hand. If the sensor reading at the lower left is greater than that on the lower right, and if the sum of sensor readings on the left side is greater than the sum of sensor readings on the right side, then the device is probably held in the left hand. This heuristic also applies for determining if the right hand is holding the device.

When picking up a device or moving it between hands, sensor readings can change drastically. In order to avoid false detection of grasp states the algorithm only outputs a guess if the same state has been recognized five times in a row. We observed a reaction time of about 600 - 700 ms for a successful detection of grasping state. The detected grasping state is shown by a simple on-screen illustration.

## USER STUDY

We conducted a preliminary user study using a repeated measures within-subject factorial design. We recruited six volunteers (one of them was female) from our institution ranging in age from 26 to 28 years (average age was 27.2). None of them had any experience with the prototype beforehand. Two of the participants were left-handed, the remaining four were right-handed.

The goal of the study was to determine the error rate of our detection algorithm as well as the detection speed. Additionally, we wanted to find possible sources of erroneous detection in order to improve the sensor layout and algorithms. As we anticipate such interactions to be conducted regularly with one's own private phone or TUI we gave all users a short introduction and training oppportunity. Users were asked to perform certain grasping actions instead of letting them choose one.

The task was to pick up the prototype from a table and grasp or hold it in one of the six ways our algorithm discriminates. Users had to perform each of the six actions ten times (in a randomized order) leading to 60 data points per user and a total number of 360 data points. Before the actual test each participant passed a training session which included five occurences for each grasping method. The way the user had to grasp the prototype was indicated by an icon on the screen. Participants were asked to switch from one way of grasping the prototype to the next one without putting the device back onto the table. This behavior mirrors real-world usage more accurately. We assume that the error rate is not lower than it would be if participants place the prototype on the table before starting a subsequent task. If a grasping method was recognized in a wrong way it has been counted as error. An error was also logged if no grasping method was recognized within three seconds. This time was measured manually, starting when the participant seemed to hold the prototype in the specified way.

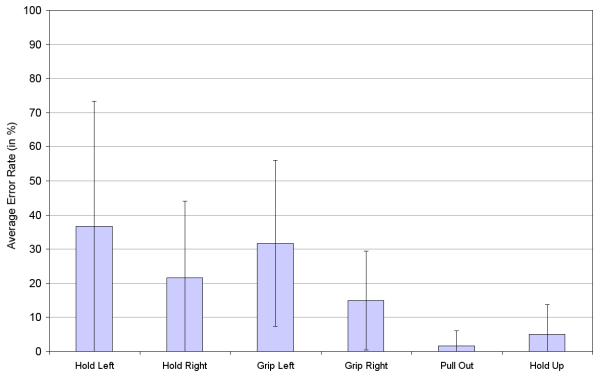


**Figure 3. Average error rate for the six grasping methods with 95 percent confidence intervals.**

In our study we didn't find any significant main effects for the grasping methods indicating that all of them work with a similar performance (see Figure 3). However, by contrast analysis we identified that the two grasping methods *pull out* and *hold up* had a significantly better performance (i.e. lower error rate) than *hold right* and *grasp left* ($F_{1,5}$=7.353, p=.42 and $F_{1,5}$=6.809, p=.48 respectively). The other two methods (i.e. *hold left* and *grasp right*) were close to have a significant difference. This indicates that detecting two fingers at a certain position denoting holding the tool up or pulling it out of a pocket seems to be easily detectable by our hard- and software. While the false detection rate for some users was very low, other users generated over 50 per cent false detections.

## APPLICATIONS

Getting reliable hints on the way a user holds a device can offer an additional channel for interacting with it. Grip information can also enhance implicit and explicit interaction. In the following we provide some examples for both areas. Especially in the research area of Tangible User Interfaces many novel hardware designs are emerging, exploring alternative ways of interacting with data. Thus, it is difficult to suggest specific enhancements that could be made using grasp-sensing surfaces. Therefore we focus our examples on well-known tangible devices like mobile phones, PDAs, and input devices. We see three main application areas of grasp

information: inferring meaning from different ways to hold an object, adjusting the user interface to the user's handedness, and enhancing spatial information using the constraints posed by grasp states and handedness.

### *Grasping States*

The way a users hold a device can be utilized for implicit or explicit interactions. An example for implicit interaction would be a phone answering scenario. A mobile phone equipped with grasp sensors would be aware of being in its owner's pocket instead of lying around somewhere. Accordingly, an incoming call would first be signalled by vibration, only after some time by a ring tone. Once the user starts pulling out the phone from his pocket, this signals that he is aware of the incoming call. Ring tone volume is turned down in order to not disturb nearby people. When the user holds the phone in his hand the display lights up so he can see who is calling him. If he decides not to answer the call he can put the phone back into his pocket where it silences. In order to answer the call he just holds the phone to his ear.

Implicit grasp information can also be used to change the behavior of a TUI depending on way it is held. For example, an interactive toy could detect whether an adult's or a children's hand touches it, presenting different tasks to them. A user could also indicate if she wants to use an input device for fine or coarse control by grasping it only with thumb and index finger or with the whole hand.

### *Handedness*

Detecting handedness of a user allows adjusting the user interface accordingly. The most obvious enhancement would be to arrange GUI widgets on a screen so that the user does not cover important screen areas when interacting with it using his hand or stylus [2]. Similarily, re-mapping hardware buttons can enhance input devices. For example, an adaptive mouse could automatically adjust its button mapping for right and left hands. Input devices could also have different functions depending on the hand they are used with. Some devices might even mechanically adjust themselves to their user's hand.

### *Spatial Hints*

Another way to exploit grasp and handedness information is to infer spatial hints from them. Often this information can be used to enhance other sensor data. For example, if a mobile phone is held in the left hand and accelerometer data shows that gravity is towards the phone's left side, it can be inferred that the user holds his phone up into the air. Proximity sensors can also tell a device whether it is held close to the body or not. When handedness and the way a user holds a device are known, it is possible for a device to determine its position relative to him. This makes certain gestures like tilting toward/away from the user possible.

## DISCUSSION

The user study showed that users hold the device in entirely different ways. This makes it hard to distinguish grasp states across different users. One way to counter this effect could be to make affordances for holding a device - for example designating possible grasping points by small depressions in the surface. During the user study we saw a learning effect. However, the study was too short to confirm it. Essentially it is not desirable that users learn how to correctly hold a grasp-sensitive device. The better solution is for the device to correctly recognize implicit ways of grasping it. However, HandSense is a proof of the feasibility of that concept. We plan to investigate the different ways users intuitively hold a device in the future. The next steps from a technical standpoint will be to decrease sensor size and embed more sensors into a tangible UI. Additionally, we will improve heuristics and increase antenna size to cover more different ways of grasping the device. This should als reduce error rates as many false detections seem to be caused by slightly 'wrong' finger positions.

## CONCLUSION

We have presented a technique for detecting how the user holds a device. This information can be used to enhance existing user interfaces but also for designing new implicit and explicit interactions. Our study suggests that handedness detection generally works with four capacitive sensors. For reliably discriminating different grasping methods more sensors are needed to cater for differences in finger position between users. We are convinced that distinguishing different grasping methods is possible and useful.

## REFERENCES


1. A. Butler, S. Izadi, and S. Hodges. Sidesight: multi-touch interaction around small devices. In *Proceedings of UIST '08*, pages 201–204, New York, NY, USA, 2008. ACM.

2. B. L. Harrison, K. P. Fishkin, A. Gujar, C. Mochon, and R. Want. Squeeze me, hold me, tilt me! an exploration of manipulative user interfaces. In *Proceedings of CHI '98*, pages 17–24, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

3. K. Hinckley, J. Pierce, M. Sinclair, and E. Horvitz. Sensing techniques for mobile interaction. In *Proceedings of UIST '00*, pages 91–100. ACM New York, NY, USA, 2000.

4. J. Mäntyjärvi, K. Nybergh, J. Himberg, and K. Hjelt. Touch Detection System for Mobile Terminals. In *Proceedings of MobileHCI '05*. Springer, 2004.

5. D. Wigdor, C. Forlines, P. Baudisch, J. Barnwell, and C. Shen. Lucid touch: a see-through mobile device. In *Proceedings of UIST '07*, pages 269–278, New York, NY, USA, 2007. ACM.

6. R. Wimmer, M. Kranz, S. Boring, and A. Schmidt. A Capacitive Sensing Toolkit for Pervasive Activity Detection and Recognition. In *PerCom '07*, Mar. 2007.