PreSense: Interaction Techniques for Finger Sensing Input Devices

Jun Rekimoto[*] **Takaaki Ishizawa**[**] **Carsten Schwesig**[*] **Haruo Oba**[*]

[*] Interaction Laboratory
Sony Computer Science Laboratories, Inc.
3-14-13 Higashigotanda
Shinagawa-ku, Tokyo 141-0022 Japan
Phone: +81 3 5448 4380, Fax +81 3 5448 4273
{rekimoto,oba}@csl.sony.co.jp
http://www.csl.sony.co.jp/person/rekimoto.html

[**] Graduate School of Media and Governance
Keio University
Shonan Fujisawa Campus
5322 Endo, Fujisawa 252-8520 Japan
taixiang@sfc.keio.ac.jp

ABSTRACT

Although graphical user interfaces started as imitations of the physical world, many interaction techniques have since been invented that are not available in the real world. This paper focuses on one of these "previewing", and how a sensory enhanced input device called "PreSense Keypad" can provide a preview for users before they actually execute the commands. Preview important in the real world because it is often not possible to undo an action. This previewable feature helps users to see what will occur next. It is also helpful when the command assignment of the keypad dynamically changes, such as for universal commanders. We present several interaction techniques based on this input device, including menu and map browsing systems and a text input system. We also discuss finger gesture recognition for the PreSense Keypad.

KEYWORDS: input devices, previewable user interfaces, gesture sensing

INTRODUCTION

As mobile devices become more common, designing effective interaction techniques for these devices becomes more important. Unlike desktop computers, input devices for mobile computers are often restricted by the limited size, and thus multiple functions must be assigned to the same buttons. This factor makes user interfaces complicated and unpredictable.

This paper proposes a new input device for mobile devices, called PreSense. PreSense is a keypad that is enhanced by touch (or proximity) sensors based on capacitive sensing (Figure 1). Since each keytop behaves as an independent touch sensor, finger motions on the keypad are recognized by integrating each keytop's information.

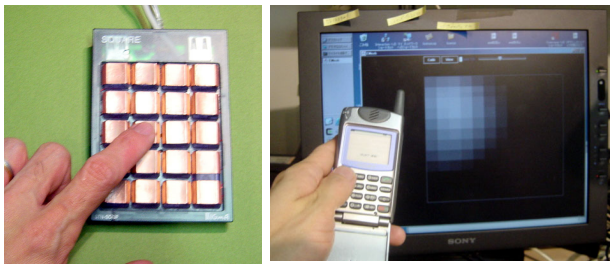

Figure 1: PreSense Keypad senses finger contact (proximity) and position before the key press.

This sensor and keytop combination makes it possible to recognize "which button is about to be pressed", and the system can provide appropriate information to users. This feature becomes effective when users are uncertain about what will occur as a result of the key press (i.e., command execution). We call it the "previewable user interface" because users can see the effect of the command in advance. Users can quickly browse multiple preview information by sliding their fingers (or thumbs) over the keypad.

The other potential benefit of the PreSense input device is gesture recognition. By estimating finger motion, or calculating the number of finger contacts, it is possible to recognize several command inputs other than normal typing.

This paper presents various kinds of interaction techniques using the PreSense Keypad. We first discuss the "preview" user interface features and then look at several concrete examples, followed by a discussion on the sensor architecture.

RELATED WORK

Our work in this paper is an extension of our previous SmartPad system [11]. This paper further enhances the concept and realizes several application examples. It also introduces different sensor designs.

Hinckley et al. proposed “touch-sensing user interfaces” [4] and developed touch-sensing input devices such as the touch-mouse. Our PreSense keypad is also a kind of touch-sensing input device and it also senses finger motion on the device’s surface by using multiple touch sensors instead of using a single sensor that covers the entire device. ThumbSense also senses finger contact on the touch-pad and uses this information to automatically change operation modes [10]. For example, when a user touches the touch-pad, some of the keyboard keys act as mouse buttons so the user can smoothly shift text input and touch-pad operation (mouse) modes without removing their hands from the keyboard’s home position.

The layered touch panel [16] is a touch-panel that is enhanced by an infrared-grid sensor. It can be used as a previewable input device because it can detect finger position before the finger makes contact with the touch panel. The pointing-keyboard [17] integrates a keyboard and an infrared-grid sensor. Our sensor is based-on capacitive sensing and the entire sensor can be embedded within a normal keypad without requiring projected sensors. This feature is suitable for integration with mobile devices.

Behind touch [5] is an input method for mobile devices that uses a keypad installed on the back of the device. Users manipulate it with their fingers and their finger position appears on the front screen.

Finally, our work was also inspired by other sensory enhanced mobile device research [8, 2, 3, 12]. While these devices use additional sensors as new (separated) input modes, we are more interested in enhancing existing input devices, such as a keypad, with sensors.

PREVIEWABLE USER INTERFACES

As Shneiderman clarified [14], the essence of direct manipulation user interfaces can be summarized as the following three features:

- Continuous representation of the object of interest.
- Physical actions or labeled button presses instead of complex syntax.
- Rapid incremental reversible operations whose impact on the object of interest is immediately visible.

The third of these defines the user interface feedback. More precisely, there are two types of *feedback*. One is where the information is delivered as a result of command execution. Sallen et al. calle this type *reactive* feedback [13]. The other type is feedback information that is given prior to command execution. This is either called *proactive* feedback [13] or more popularly, *preview*. [15]

A simple example of preview information is a command tooltip (Figure 2). When the cursor stops at one of the tool button icons, more detailed information than simple the button label pops-up on the screen. Users know which button must be pressed before executing all the buttons. The Tooltip

Figure 2: Preview information example: a tool tip automatically pops up when the cursor stays over an icon.

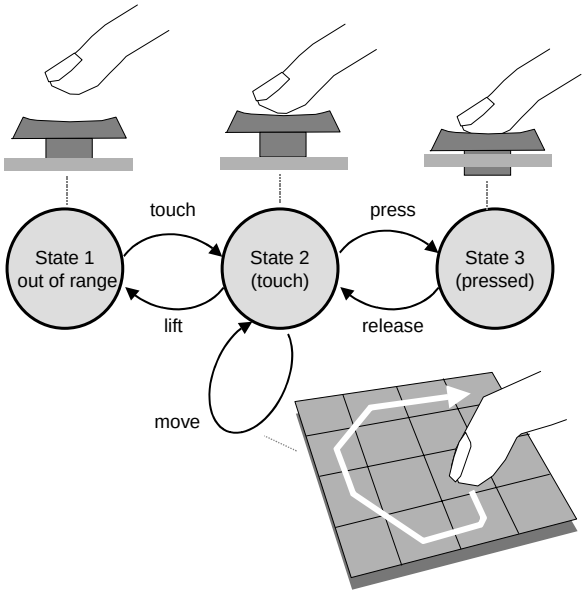

Figure 3: Three input states of PreSense: the “touch” state can be used to provide preview information before command execution. Finger movement without pressing the buttons can be treated as gesture commands.

information automatically disappears after the cursor goes away.

Although (reactive) feedback and preview are supplemental concepts and can be used in combination, “preview” has several features that are not available with normal feedback, as demonstrated in this simple tooltip example. These features can be summarized as the following:

Transparent operations: In many cases, interacting with preview information, such as tooltips, is transparent. It does not force users to change operation. Experienced users can simply bypass previews and directly perform commands, while the same system can guide novice users.

Rapid inspection of possibilities: When users do not understand the command results well, one possible solution is to execute one of them and undo it if the result does not match their expectations (and then try another). However, this approach is often ineffective because of the cognitive overload of repeating “do-undo” operations, and also because not the all commands are undo-able.

Sensor modality	Visual	Auditory	Kinesthetic
Timing of delivery *1	Reactive	Proactive (previewable)	
Duration *2	Transient	Sustained	
Monitoring *3	Demanding	Avoidable	
Mode onwer *4	User-maintained	System-maintained	

*1: Does feedback occur only when an action is executed?
*2: is the feedback sustained throughout a particular mode?
*3: Can the user choose not to monitor the feedback?
*4: Does the user actively maintain the mode?

feedback styles with PreSense

Table 1: Feedback styles (based on Sellen et al.) and PreSense.

The preview information assists users in selecting the appropriate command before actually executing it. It can be either real (e.g., the result of a font style change can be previewed in a popup window) or modeled (e.g., showing a print image before sending a document to the printer).

This "undo-less" feature is particularly beneficial when users' tasks affect real world situations (e.g., printing out a document), because the results of the executions are often difficult or impossible to cancel.

Efficient use of screen space: Techniques such as tool-tips can be regarded as an effective way of using (limited) screen space because guide information is selectively presented according to the user's current focus. If all the information appears on the screen, it would become too complicated and would also consume a large amount of screen area. When users are dealing with mobile devices, the "context sensitive information presentation" strategy would be very important because the screen size is more restrictive than in desktop computing.

Previewable Interfaces with Physical Keypads

For these reasons, the "preview" techniques effectively help users in various GUI applications [15]. However, when dealing with physical input devices, such as a keypad of a cellular phone, "preview" was not effectively provided.

One reason for this is that while the mouse (and electro magnetically traced stylus devices, such as those made by Wacom) can express three states. However, as Buxton explained [1], many important GUI techniques require three input device states, such as out of range, staying on a target without pressing the button, and actual button press. Normal physical buttons only have two input states (i.e., out of range and pressed) and thus it is not possible to implement all the GUI techniques, including preview.

This problem can be addressed by introducing a sensor-enhanced keypad, called the "PreSense keypad", that can sense a user's finger position on the keypad without the buttons being pressed (Figure 3).

Figure 4: Map navigation with PreSense Keypad. The area corresponding to the user's thumb position becomes slightly larger than other areas, and the property information also becomes visible. Pressing the key zooms-up the corresponding area.

When a user first puts his/her finger on the keypad, preview information is provided on the screen. The information can be modified according to the finger position on the keypad. Then, the user can actually execute the previewed command by physically pressing one of the buttons. Although a similar interface can be implemented by a combination of arrow keys and a selection button, our combination of finger motion and key pressing is much faster than those alternatives. This interaction is also "transparent" because users can easily ignore the preview information once they are accustomed to the system by directly pressing one of the keys.

Adding to this feature, finger motions on the keypad without pressing the buttons can also be recognized as commands.

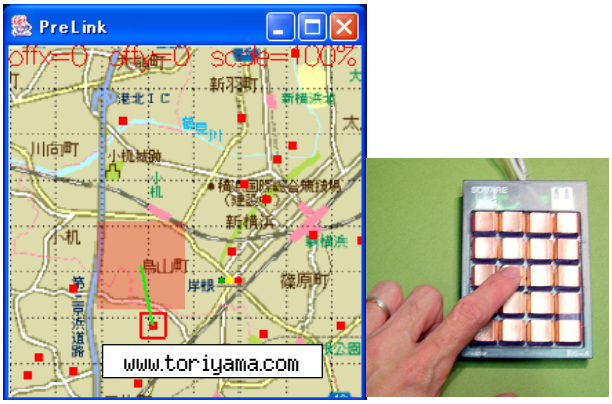

Figure 5: A link navigation example. Link points are on the map surface, and users can select one of these links by sliding the thumb over the keypad. The selected link is highlighted and can provide additional information such as a link name or a URL, before the link information is actually retrieved.

This feature enables gesture interactions to be combined with previewable interfaces.

Sellen et al. discuss effective feedback styles for avoiding mode errors [13]. They also classify various aspects of feedback, as shown in Table 1. This table also shows possible feedback styles that can be provided with PreSense. According to Sellen et al.'s experiment, the frequency of mode error reduces when the system can provide kinesthetic feedback and when users can control modes. PreSense satisfies these conditions. It can provide kinesthetic feedback because users can feel the button shape when touching it.

EXAMPLES OF PHYSICAL PREVIEWABLE INTERACTIONS

This section examines various examples of previewable interfaces with the PreSense keypad.

Map Navigation

One possible use of preview is to assist mobile device users in browsing large information spaces, such as maps. Since the display size of mobile devices is often limited, an effective navigation technique, such as zooming, is very important. Mobile devices also often lack two-dimensional pointing devices, and selection of the area to zoom is not a trivial task. One possible way is to use arrow keys, but this requires several key taps. Another recently proposed technique is PartNavi [6], where the screen area is divided into grid sections which are the same rows and columns as the keypad. Users can directly select one of these sections by directly pressing the corresponding key on the keypad.

Our zooming technique uses grid-shape sub areas and the system highlights the selected area when users touch a key on the keypad (Figure 4). With this preview information they can intuitively understand the relation between the keypad and screen areas. Pressing one of the keys zooms up the associate region on the map.

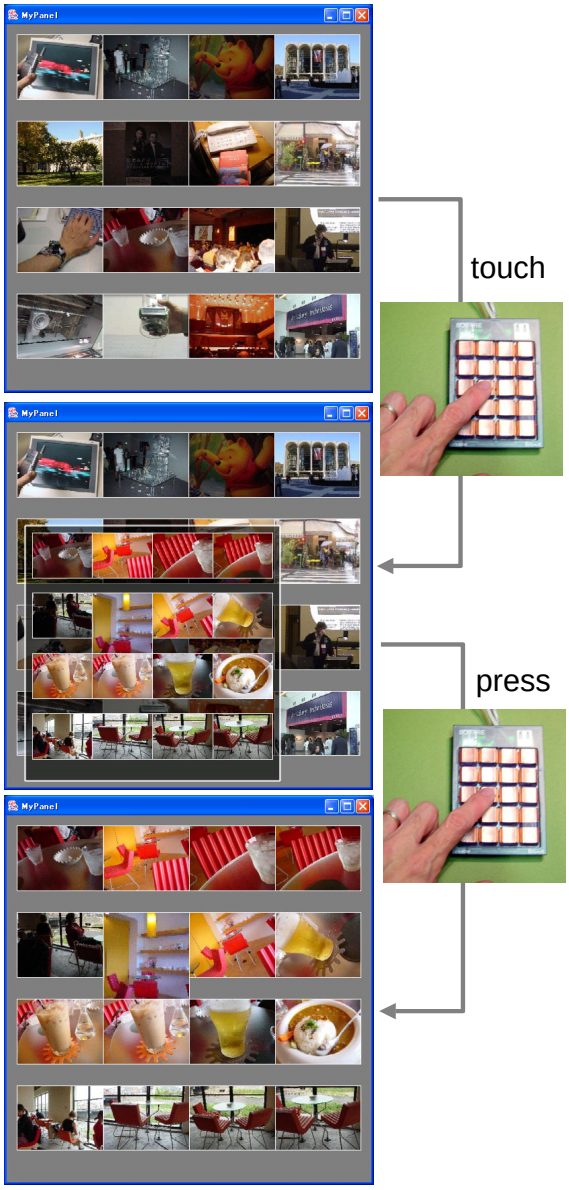

Figure 6: Hierarchical object selection example: touching one of the keytops pops up the associated next level information. A user can inspect information one-level ahead before opening it.

This zoom interface can also be combined with simple gesture commands. For example, touching the two separate keycaps brings the zooming state to the previous level (i.e., zoom down), and contacting the keypad with a palm (which causes three or more contact points) resets the zooming to the initial state. We will see other variations of gesture commands in a latter section.

Link Traversal

Another example is link selection. For example, Figure 5 shows a map with several anchor points and these links are associated with URL links. When users put their fingers over one of the keys, the corresponding area is highlighted and the

Figure 7: A universal commander example: A user controls devices with a single remote commander. The command assignments might be dynamically changed based on the selected target device. Preview information on the display (either on a remote commander's small screen or the target device's screen) frees users from remembering all the key combinations of different types of target devices.

links included in this area are also highlighted. If the number of such links is one, pressing the same key can invoke the link (e.g., open an associate web page, etc.). If the number of links is more than two, the users first press the key to enlarge that area, and then repeat the link selection operation.

Looking ahead to the next level

Another possible example is hierarchical information browsing, such as finding a document from folder trees, or selecting a command from menu hierarchies. First, the system shows the top-level items on a display. The layout of these items are configured the same as the configuration of the keypad. To inspect the next level, users can touch one of the keys and the next level items associated with the selected key pops up on the screen (Figure 6). By sliding a finger over the keypad, a user can quickly scan the next level items of different current-level items.

For example, when navigating document folders, icons contained in the folder can be seen before actually opening them, This feature eliminates unnecessary open-and-close operations when a user is looking through the information hierarchies.

Figure 7 shows a different situation where a user is selecting the chosen item by sliding his/her finger over the keypad. In this case, detailed information associated with the selected item pops-up on the screen and the user commits the selection by physically pressing the button.

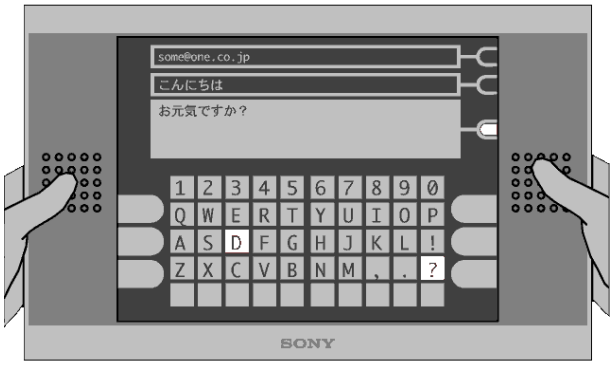

Figure 8: A split keyboard design and previewable keypad. Software keyboard automatically appears on the screen when a user's fingers make contact with the pad.

Figure 9: Users can select the vertical or horizontal styles according to the situation. Preview information on the screen helps a user recognize dynamic keypad modifications and command bindings due to the rotation of the keypad.

TEXT INPUT WITH PRESENSE KEYPAD

Another area of interaction techniques where PreSense becomes useful is text input.

Figure 8 shows a combination of the on-screen keyboard and small physical keypad. The user can check the corresponding character on the screen before typing it. Even if the keytops are not large enough to print the key labels, the screen preview information can assist users to confirm the entered characters. This feature can also support multiple function keypads. For example, when a mobile device is used as a universal remote commander, the keypad functions will change according to the current operation mode. Using this information, users can quickly confirm what is assigned to a particular button before pressing it. This is also a variation of "preview" interfaces.

Similar to the "on-demand interface" [4], the on-screen keyboard dynamically appears only when users touch the one of the keypads. This technique makes effective use of the screen space without requiring explicit mode changing commands.

The other potential technique is to provide two-way keypads. Figure 9 shows this concept. Users would be able to use a keypad in two styles (vertical and horizontal), according to

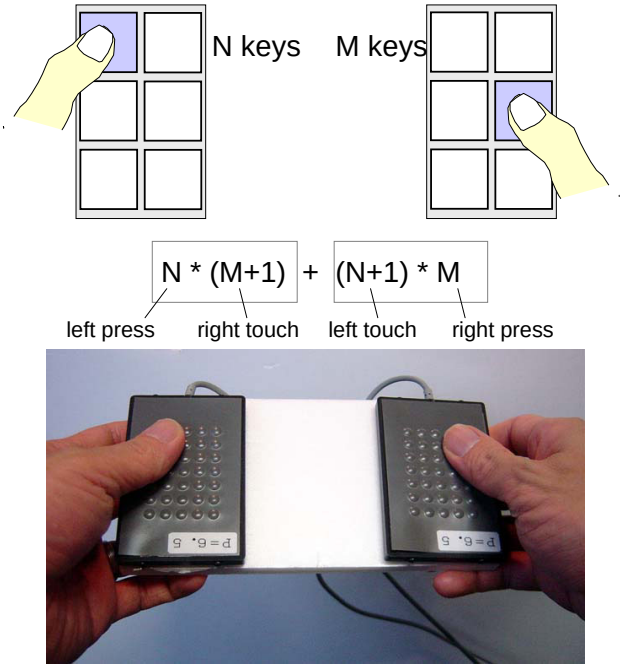

Figure 10: The number of characters that can be distinguished with split touch-shift keypad.

the particular application on the cellular phone. Since the keytop information can be dynamically shown on the screen, users do not have to rely on the physical keytop labels.

Multi-Key Input

The PreSense technique can also be applied to multi-key input methods. A common technique for entering text with a limited number of keys is to use a combination of keys. Key combinations might be sequential (e.g., a key press followed by another key press), or chorded (e.g., pressing two keys at the same time).

More specifically, if two keys are used to represent one character, and there are N keys, then $N \times N$ characters can be represented using such combinations. Ideally, such multiple key methods can handle a number of characters with a small number of keys. However, in practice two problems would arise.

Learning key combinations: Users must learn all the key combinations. If there are not good assisting technologies provided that help users to learn key combinations, the learning curve would be steep and users could start using them only after a certain period of practice time.

Delimiting ambiguity If the keypad takes a sequence of multiple key taps, separating multiple strokes is not always clear. For example, if the key combinations "A-B", "B-C", and "C-D" etc represent corresponding characters, and the intention is to type "A-B, C-D, E-F", but the user happens to drop the first key ("A"), the resulting sequence becomes "B-C, D-E, F-G" and a completely different character would be entered.

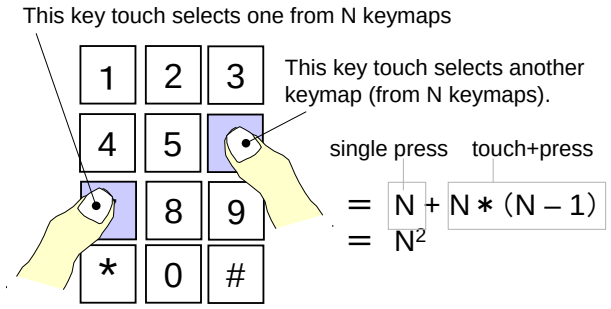

Figure 11: Alternative design of the Touch-Shift keypad. Key touches by one-finger selects key map for the key press.

Touch-shift

Our proposed solution is to use a touch-sensitive keypad for text entry. If each key has touch sensitivity, three input states (not-contact, touch, and press) can be distinguished. Thus, there is the possibility of distinguishing more text characters with a limited number of keys compared with the traditional keypad where each key only has two states. We call this input method the "touch-shift" input.

One possible design of touch-shift is to use two separate keypads for both thumbs and represent a character by a combination of key touch and key press. For example, if a user touches one of the left key tops, an associated keymap for the right keypad is selected. Similarly, the right key touch determines the left keypad bindings.

Suppose that we have N and M keys on the left and right keypads respectively. There are $N + 1$ touching states for the left keypad (touching N keys + no touch) and thus there are $(N + 1) \times M$ ways to press the right keys. In the same manner, there are $N \times (M + 1)$ ways to press the left keys, and the total number becomes $((N+1) \times M) + (N \times (M+1))$ (Figure 10).

For example, if the right part is compatible with a phone keypad (12 keys), and the left keypad has 4 keys, the total number would be 112 ($= (5 \times 12) + (4 \times 13)$), and this number covers all the printable ASCII characters.

Another possible keypad design is to use a single keypad and combination of two key touches. Figure 11 shows the keypad and the corresponding visual guide on the keypad. Using this combination, one (e.g., left) key touch decides key binding for the second (e.g., right) key press, and vice versa. The total number of selectable characters is $N + (N \times (N-1)) = N^2$. For example, a 12 keysphone keypad would handle 144 characters.

For this two-thumb keypad, preview information can be provided as shown in Figure 12. Each key top on the screen shows two character labels, corresponding to two keymaps selected by each thumb touch. As the user slides their thumb over the keypad, different keymaps are selected and the pre-

Figure 12: Preview information for the touch-shift keypad. Two character labels on each keytop represent two selected key layouts. Each is selected by touch.

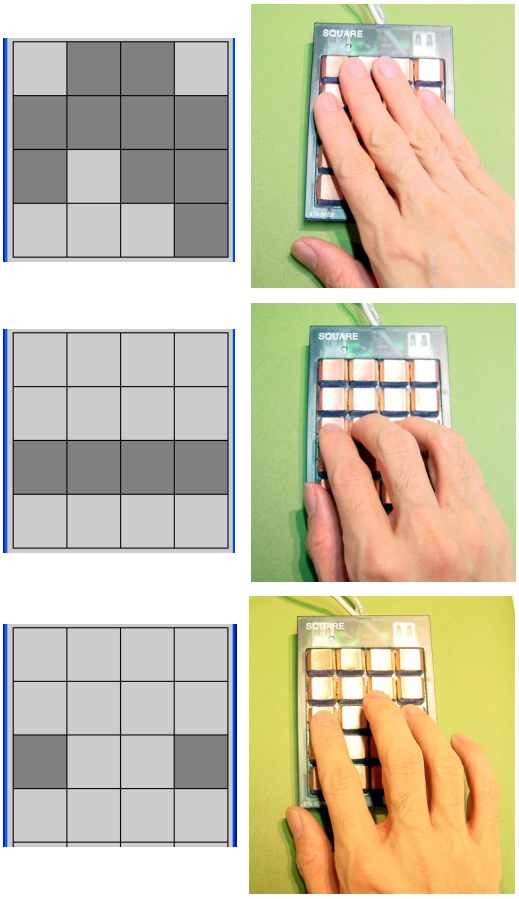

Figure 13: Putting multiple fingers makes different touch-patters from normal key pressing, thus can be treated as different input commands.

view information changes accordingly.

Although formal evaluation has not been conducted, our initial experience with the tentative multi-key definition showed that this method really helps users to incrementally learn new key assignments. Since one of the most serious burdens that inhibits normal users from using a chorded keyboard is the learning problem, the combination of touch-shift and dynamic keytop preview can provide a new solution for those input methods. To test this assumption, we are currently planning to apply this technique to chording key methods designed for stenography keyboards.

GESTURE INTERFACES

In the previous sections, we have seen various examples of preview interfaces with the PreSense keypad. Apart from the preview, the other general area of interaction techniques of PreSense is gesture input. That is, finger motions or ways of touching the keypad can be recognized as different input commands, in addition to normal key inputs.

We are currently investigating two groups of gestures. The first category is gestures that can be combined with normal key inputs. Gestures of this group must be distinguishable from finger motions for key inputs. The other category assumes an explicit gesture mode.

Tapping and Holding

The simplest gesture is tapping the keypad. We have already seen several previewable interfaces that are activated by contacting the keypad and touching can be regarded as the simplest gesture command. In addition, some techniques that are designed for the touch-panel or the touch-pad can also be used. For example, “tap and hold” (touch-and-release within a certain time period, and then touching the keytop again) can be used as another command.

Contact patterns

Another category is the placing of the fingers on the keypad. For example, putting a palm on a keypad results in multiple contact and thus can be treated as a special command because it is easily distinguishable from normal typing (Figure 13 above). Such gestures can be used to reset the system status (e.g., while navigating the menu hierarchy, a user can immediately return to the root menu just by putting his/her palm on the keypad). Another example of a static gesture is to put multiple fingers on the keypad. For example, touching with three fingers causes an associated command (Figure 13 middle), or touching two keys on the left and right edge can

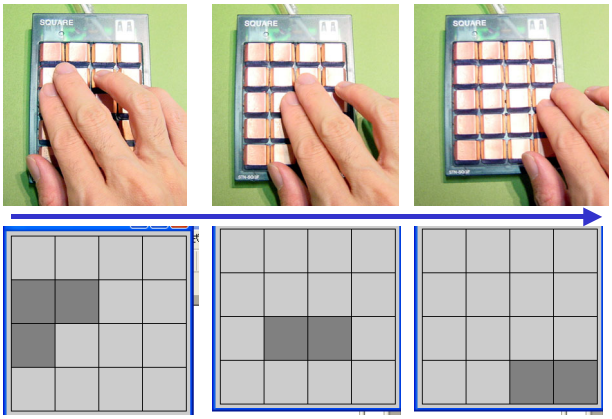

Figure 14: Wipe motion on the keypad

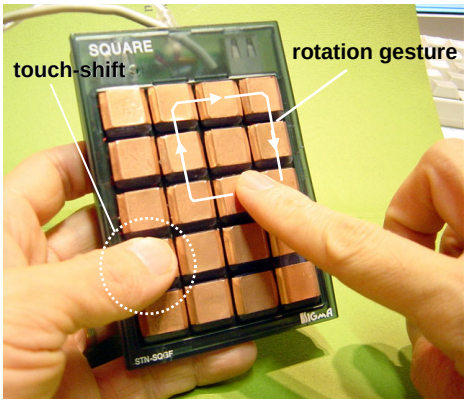

Figure 15: Touch-shift and gesture commands: a round motion of the index finger becomes a virtual "jog" dial.

be used as a gesture command.

Finger Motions

The other types of gestures are "dynamic" ones. For example, a "wiping" motion on the keypad turns the page on a screen (Figure 14). We use two techniques to distinguish these gesture motions from those caused by normal key inputs.

The first is to use the number of contact points. For example, the wiping gesture in Figure 14 is recognized only when two or more fingers touch the keypad.

The other technique is to use the motion speed. For example, when a finger touches a row of keytops from bottom to top, with a speed of more than the predefined threshold, this motion is treated as a gesture command.

Gestures with explicit mode shifts

The above examples are gesture techniques that can be mixed with normal keypad operations (i.e., typing). However with the explicit mode, several finger movements can be treated as various gesture commands. There are several ways to specify gesture modes: key commands or combinations, or one of the

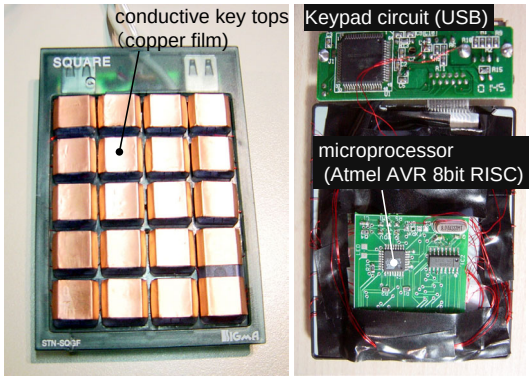

Figure 16: A keypad with a touch sensor on each keytop.

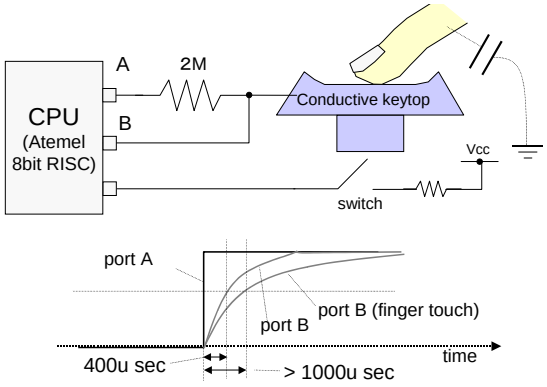

Figure 17: Configuration of the touch-sensitive key top. Both A and B are the digital IO ports (port A is output and port B is input).

above gestures can be used.

The other way is to use "touch-shift", as we have seen in the text input section. For example, touching one key (the gesture-shift key) with a finger movement on another key is handled as a gesture command. We implemented this technique for the map navigation system. A user can zoom in by pressing one of the keytops and can also scroll the map by moving the finger over keypad while touching the "map-scroll" key.

Using this touch-shift method, more complicated (and thus difficult to distinguish from key input motions) gestures become possible. For example, a circular motion on a keypad can be used as a "virtual" jog dial.

SENSOR ARCHITECTURE

To enable the interactions described in the previous sections, the device must sense the finger position and contact information. The PreSense keypad uses capacitive sensing for this purpose. We are currently investigating three sensor designs, and planning to implement some of the other configurations.

Keypad + touch sensors: The first sensor configuration connects a touch sensor to each key (Figure 16). Figure 17

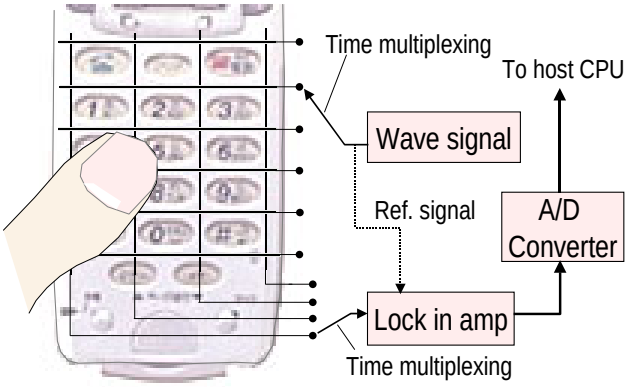

Figure 18: An alternative sensor configuration based on SmartSkin: An arrayed electrode sensor detects the finger position on the keypad. (Note: electrodes are embedded in the keypad.)

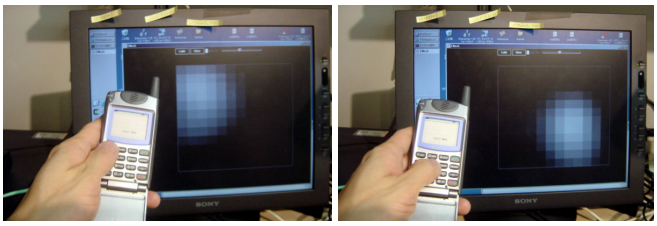

Figure 19: A prototype cellular phone keypad with the PreSense sensor. The recognized finger position appears on the screen.

shows the circuit required for each key. This sensor measures the delay time of the capacitor-resistor (CR) circuit. When the finger touches the (conductive) key top, its capacitance increases and thus affects the measured delay time. Three lines (two for measuring contact sensing and the other line for detecting normal key switch) are required for each keytop, and the Atmel AVR 8bit RISC microprocessor can sense up to 25 (5×5) key tops. Since each keytop behaves as an independent touch sensor, calculating multiple finger contact is possible. Our current prototype can scan all 25 key tops 50 times per second.

SmartSkin sensor: We are also investigating the alternative sensor configuration shown in Figure 18, which is based on SmartSkin [9]. In this design, a grid of electrodes is embedded on the keypad surface (i.e., between keypad buttons), and each electrode intersection acts as a capacitive proximity sensor. The system interpolates these proximity values and detects the finger position. Figure 19 shows the obtained sensor value (appeared on the screen) and corresponding finger position. The potential of this sensor configuration is that it does not require strict contact to the keytops. As shown in Figure 18, the position of the finger is detected from the electrodes that are attached to the keypad cover back. On the other hand, determining the finger position is rather unstable and the measured position shifted when the finger leans. We are currently developing a calibration method that can compensate for this position shifting.

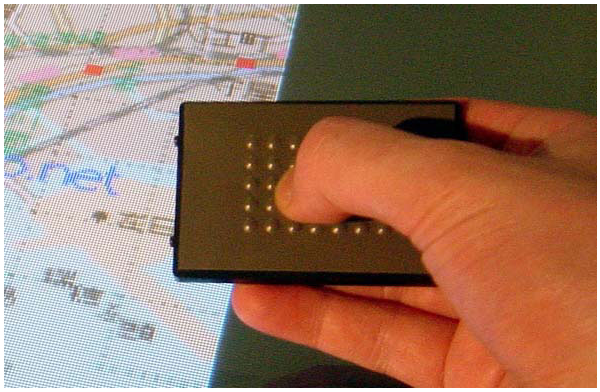

Figure 20: A touch-pad with back-patched push buttons.

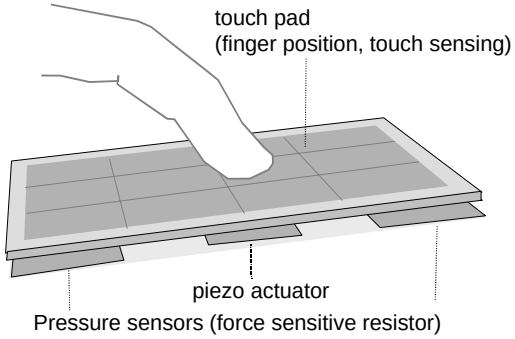

Figure 21: Pressure-sensitive touch-pad configuration. Button press feeling is generated by a piezo-acutuator.

Touchpad + pushbutton array: The third sensor design is based on a modification of the touch-pad, a popular input device for notebook computers. The touch-pad senses the position of the finger on the sensor pad by measuring capacitance change on the pad surface. It also senses finger contact (without motion) on the pad (i.e., it works as a touch sensor).

This sensor configuration combines the touch-pad with an array of small push buttons (tact switches) (Figure 20). These buttons are placed behind the touchpad sensor surface. The user can operate this pad just as a normal touchpad, and can also feel the button positions because the area of the buttons is slightly bulged. Finger position information is used to identify the button when a user touches it without pressing.

Using this configuration, multiple finger detection is not possible because of the limitation of the touch-pad sensor. Therefore some of the user interaction techniques described in this paper (such as multiple finger gestures) are not available. Nevertheless, this configuration is attractive because it can add PreSense functionality to the widely used input device.

Other possible sensor configurations: In addition to these implemented sensor configurations, we are also considering other possibilities.

One example is a combination of a normal touch pad with the pressure sensor (Figure 21). This is similar to the third implementation, but without physical buttons. Instead, finger pressure can be detected and when the given pressure exceeds a certain threshold, it is treated as a "press" operation (note that finger "touch" can be sensed by the touch-pad). Although the pressure is given to the entire input device, the system can realize multiple virtual buttons by combining pressure information with finger position information that is obtained by the touch-pad. To provide physical "click" feedback, a piezo-acutator [7] is added to the input device.

CONCLUSION AND FUTURE WORK

This paper presented PreSense, a new input device that combines a physical keypad with a position sensor. We are currently developing various applications using this device, implementing different key/sensor configurations (e.g., one-dimensional line of buttons).

Although our early implementation of touch-shift text input shows potential, the formal usability tests are yet to be conducted. We are very interested in measuring the quantative performance of the touch-shift in terms of input speed, error ratio, and learning curve.

We are also interested in designing other keytop layouts, especially for small remote commanders for portable audio devices. For example, even when a very limited number of buttons, such as two touch-sensing keytops are used, it would provide a rich selection of gesture commands (e.g., wiping two buttons, touching and holding one key then sliding toward the other, or touching two buttons at the same time). The arrow key could be enhanced in a similar way. Our next step would cover a wide variety of interaction techniques with a limited number of, but sensory enhanced, physical interactive objects.

Acknowledgements

We thank Tatushi Nashida for the valuable advice and suggestions on this work.

REFERENCES

1. W. Buxton. A three-state model of graphical input. In *Human-Computer Interaction - INTERACT'90*, pages 449–456. Elsevier Science Publisheres B.V., 1990.

2. Beverly L. Harrison, Kenneth P. Fishkin, Anuj Gujar, Carlos Mochon, and Roy Want. Squeeze me, hold me, tilt me! an exploration of manipulative user interfaces. In *CHI'98 Proceedings*, pages 17–24, 1998.

3. Ken Hinckley, Jeff Pierce, Mike Sinclair, and Eric Horvitz. Sensing techniques for mobile interaction. In *UIST 2000*, pages 91–100. ACM, 2000.

4. Ken Hinckley and Mike Sinclair. Touch-sensing input devices. In *CHI'99 Proceedings*, pages 223–230, 1999.

5. Shigeo Hiraoka, Isshin Miyamoto, and Kiyoshi Tomimatsu. Behind Touch: A text input method for mobile phone by the back and tactile sense interface. In *INTERACTION 2003, Information Processing Society of Japan (in Japanese)*, pages 131–146, 2003.

6. Haruka Kataoka and Takeo Igarashi. PartNavi - object selection by recursive screen partition. In *INTERACTION 2003, Information Processing Society of Japan (in Japanese)*, pages 61–62, 2003.

7. Ivan Poupyrev, Shigeaki Maruyama, and Jun Rekimoto. Ambient touch: designing tactile interfaces for handheld devices. In *ACM UIST 2002*, pages 51–60, 2002.

8. Jun Rekimoto. Tilting operations for small screen interfaces. In *Proceedings of ACM UIST'96*, pages 167–168, 1996.

9. Jun Rekimoto. SmartSkin: An infrastructure for freehand manipulation on interactive surfaces. In *CHI 2002 Proceedings*, pages 113–120, 2002.

10. Jun Rekimoto. ThumbSense: Automatic input mode sensing for touchpad-based interactions. In *CHI 2003 Late braking*, 2003.

11. Jun Rekimoto, Takaaki Ishizawa, and Haruo Oba. SmartPad: A finger-sensing keypad for mobile interaction. In *CHI 2003 Late braking*, 2003.

12. Carsten Schwesig, Ivan Poupyrev, and Eijiro Mori. Gummi: User interface for deformable computers. In *to be presented at CHI 2003 Interactive posters*, 2003.

13. A. Sellen, G. Kurtenbach, and W. Buxton. The prevention of mode errors through sensory feedback. *Journal of Human Computer Interaction*, 7(2):141–164, 1992.

14. Ben Shneiderman. Direct manipulation: A step beyond programming languages. *IEEE Computer*, 16(8):57–69, 1983.

15. Michael Terry and Elizabeth D. Mynatt. Side Views: Persistent, on-demand previews for open-ended tasks. In *Symposium on User Interface Software and Technology (UIST'02)*, pages 71–80, 2002.

16. Yujin Tsukada and Takeshi Hoshino. Layered touch panel: the input device with two touch panel layers. In *CHI 2002 Interactive Poster*, pages 584–585, 2002.

17. Yujin Tsukada and Takeshi Hoshino. PointingKeyboard: An input device for typing and pointing. In *Workshop on Interactive Systems and Software (WISS 2002) (in Japanese)*, pages 61–66, 2002.