Technology Reports

Grip Feature UI Smartphone

User Interface Using Natural Gripping Features —Grip UI—

Communication Device Development Department Masakatsu Tsukamoto[†]
Yuta Higuchi
Takashi Okada

The popularity of touch panels as a UI has led to larger screens and terminals in recent years, but cases in which such larger devices are felt to actually hinder operability are on the increase. Focusing on the commonplace action of "holding a smartphone," we have developed a prototype UI that recognizes each user's natural way of gripping a terminal and makes one-handed operation easier to perform. This article describes the basic concept of this novel UI and its implementation in a prototype terminal.

1. Introduction

Intuitive operations based on the touch panel have been well received in recent years and the use of smartphones has been growing explosively as a result. This trend has been accompanied by larger displays and higher resolutions as well as larger terminal enclosures. However, larger devices can effect the ease of holding a terminal and performing operations, and as a result, one-handed operation has become more difficult as certain points on the screen become more difficult to reach and touch. Nevertheless, there are many scenarios in which one-handed operation is unavoidable, such as when standing in a crowded train, carrying many personal belongings, and holding a child's hand. Having to perform such one-handed operations is increasingly causing users to drop their terminals or touch the wrong functions on the screen, and this is making users increasingly dissatisfied with smartphone operations (**Figure 1** (a)).

Of course, downsizing the terminal could dispel this dissatisfaction, but the advantage of enjoying data-intensive content on a large screen would be lost. Various methods have been tried to resolve this issue such as improving the User Interface (UI) by rearranging buttons or adjusting touch operations for one-handed operation or by incorporating motion-based operations such as shaking the terminal. These methods, however, have not been able to sufficiently improve usability for various reasons such as limitations caused by the range of finger motion or crowded conditions and the need for compatibility with two-handed operation. There have also been studies on changing the way functions are initiated according to the way in which the terminal is being held [1] [2], but they have not led to improvements in one-handed operation.

Against the above background, we have focused our attention on the commonplace action of "holding (gripping)

† Currently, Research Laboratories

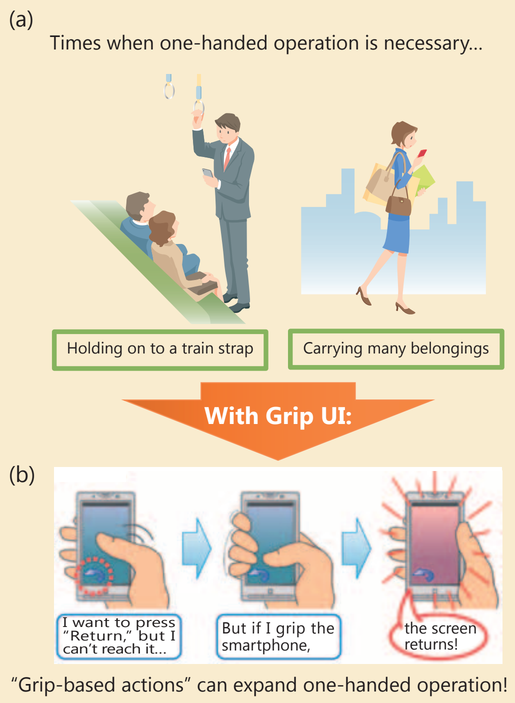

(a)
Times when one-handed operation is necessary...
Holding on to a train strap
Carrying many belongings
With Grip UI:
(b)
I want to press "Return," but I can't reach it...
But if I grip the smartphone,
the screen returns!
"Grip-based actions" can expand one-handed operation!

Figure 1 Use cases

a smartphone" and have developed a UI that can recognize each user's way of gripping a terminal (grip features) so that gripping can be used to operate a terminal (Fig. 1 (b)). This novel UI improves one-handed operation without having to change the way in which a terminal is held or to worry about crowded conditions.

In this article, we define the grip features used by this UI (hereinafter referred to as "Grip UI"), present requirements for terminal implementation, and describe the technical issues that must be addressed to satisfy those requirements and the techniques for resolving them. We also introduce a prototype terminal that we developed as an example of achieving a Grip UI.

2. Grip UI Requirements and Specifications

2.1 Definition of Grip Features

To exploit grip features as a UI, we need to quantify the state of gripping a terminal. Thus, at the beginning of this development project, we decided to define grip features that we deemed necessary for detecting the state of gripping. Grip features, however, are dynamic in nature, and we studied them knowing that they are not uniform across all individuals.

In detecting a state of gripping, it is necessary to determine "which parts" of the terminal are being gripped and "how intensely" those parts are being gripped. Among these, grip intensity can be detected on the basis of "pressure intensity." However, detecting which parts are being gripped is more problematic as that depends on the shape of the terminal and individual skeletal features. Furthermore, considering that force is not applied uniformly to the terminal, it is difficult to accurately detect which parts are being gripped solely on the basis of a "contact area" index. It is therefore important that pressure bias be obtained in a more detailed manner through a "pressure distribution." It must also be kept in mind that a terminal is usually being held by the user, and to therefore determine under what conditions the terminal is being gripped, "pressure time transition" also becomes a necessary feature in using a Grip UI.

For the above reasons, we define grip features in terms of the three feature quantities listed below (**Figure 2**).

(1) pressure intensity
(2) pressure distribution
(3) pressure time transition

(1) The "pressure intensity" feature is needed to determine differences in the pressure applied to the smartphone to judge whether force is being applied intentionally when holding the smartphone.
(2) As the name implies, the "pressure distribution" feature is needed to obtain the distribution of pressure on the terminal. The points at which pressure is being applied depends on individual characteristics such as

hand size, shape and thickness. Obtaining the pressure distribution on the terminal can therefore absorb individual differences.

(3) The "pressure time transition" feature is needed to obtain temporal variation in detected pressure. "Gripping" is a dynamic action, and even the order of fingers used to apply pressure differs between individuals. In addition, assessing the temporal continuity in pressure detection makes it possible to judge whether that pressure is intentional or accidental.

We consider that defining the above three feature quantities will enable us to obtain the information needed to achieve a Grip UI.

2.2 Terminal Implementation Requirements

The Grip UI is a function for performing terminal operations according to detected grip features. The requirements listed in **Table 1** must be satisfied for terminal implementation of Grip UI.

Items 1 and 2 in the table are requirements concerned with hardware. Development of Grip UI began with the aim of improving user friendliness assuming the use of smartphones, which means that the size of the terminal enclosure must not change to the point of changing the user's sense of terminal manipulation. At the same time, sensors that can obtain the three feature quantities defined in section 2.1 regardless of user attributes (age, gender, physique, etc.) must be implementable without affecting terminal shape.

Items 3 to 5 in the table are requirements concerned with software. Although Grip UI is to be implanted in terminals as a new type of UI, the conditions under which it can be useful differ from those of existing UIs, and for this reason, it should be used in parallel with existing UIs such as touch panels and hard keyboards. In other words, Grip UI must be able to coexist with existing UIs. Furthermore, as a UI is a function related to all applications, a framework supporting Grip UI must be prepared in such a way that applications do not have to be modified and the extendibility of existing applications and 3rd Party[*1] applications can be guaranteed. Additionally, as grip features themselves are physical quantities obtained by sensors, there is no reason to believe that they cannot be beneficially used for other applications in addition to terminal operations. This kind of implementation that can ensure

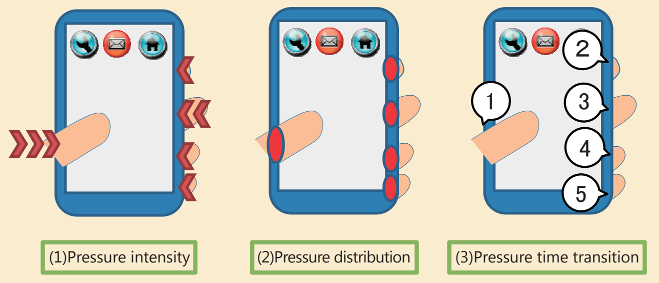


Figure 2 Requirements for detecting a gripping state

Table 1 Performance requirements of Grip UI

No.	Item	Performance requirement
1	Hardware	Grip features must be obtainable regardless of user attributes (age, gender, physique, etc.)
2		Must not be larger than existing terminals to the point of changing the way a terminal is held.
3	Software	Must not conflict with existing UIs such as touch panels.
4		Must be usable without having to modify applications.
5		Detected grip features must be applicable to other than terminal operations.
6	Usability	It must be possible to register and use each individual's style of gripping.
7		It must be easy to register a gripping style.
8		Users must be able to sense execution by Grip UI.
9		Users must not feel that response time is slower than that of existing UIs.
10		Basic performance of existing terminals must not be lost.

*1 **3rd Party**: A software vendor that develops software for UE.

such extendibility is desirable.

Items 6 to 10 are requirements concerned with using Grip UI to perform terminal functions. Items 6 and 7 are requirements for making new functions easier to use while items 8 and 9 are requirements for providing users with a compelling and enjoyable interface experience.

Finally, item 10 is a requirement that can be treated as a precondition to satisfying items 1 to 9. Given that the purpose of this development project is to improve the basic performance of the terminal (UI), ensuring that no functionality is lost as a tradeoff is an important study item.

3. Grip UI Functional Implementation

3.1 Optimization of Tactile Sensors

In this development, we decided to use a tactile sensor [3]–[5] capable of multipoint pressure measurements with the aim of efficiently detecting “pressure intensity” and “pressure distribution” from among the grip features described in section 2.1. Specifically, to achieve an effective multipoint measurement system, we used a tactile sensor that configures pressure sensor elements in an array[*2] shape.

A key characteristic of the tactile sensor used here is that making the area of a single pressure sensor element larger enables a lower pressure intensity to be detected and a high-sensitivity implementation to be achieved. On the other hand, it is desirable that the area of a single pressure sensor element in an array be small to obtain a detailed understanding of pressure distribution. With the above in mind, we worked on optimizing the size and the array of the pressure sensor elements making up the tactile sensor (**Figure 3**). Since one requirement in this development is that Grip UI is not to influence terminal shape, we decided to adopt a sheet-shaped tactile sensor. This approach satisfies the hardware requirements presented in section 2.2.

To optimize an actual arrangement, we performed evaluation experiments repeatedly on prototype devices using the individual size, number and location of pressure sensor elements making up the tactile sensor as parameters. These experiments revealed that an increase in elements increased the load applied to the A/D converters and control ICs needed for obtaining sensor information. One measure commonly used to reduce current consumption is to lengthen the sensing interval, but in this optimization, we also had to ensure that the “pressure time transition” feature described in section 2.1 could also be sufficiently detected.

3.2 Coexistence with Existing UIs

On incorporating a new sensor, it must be interconnected with Android™[*3] OS without affecting existing functions. Since a new UI is implemented through interrupt processing on Android OS, placing the new sensor on the same level as existing UI devices may generate competition and degrade usability.

To resolve this issue, we used a basic framework for sensor addition prepared by Android OS as shown in **Figure 4**. Specifically, we added a tactile-sensor interface in the extension area provided as standard in the SensorManager[*4] block implemented in the application framework[*5]. Thus, from the viewpoint of Android OS, the new sensor took on a form equivalent to that of an accelera-

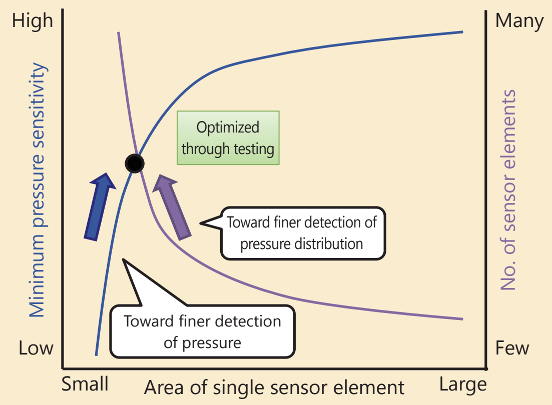


Figure 3 Sensor size versus sensor characteristics

[*2] **Array**: The arranging of sensor elements in an array-shape configuration.

[*3] **Android™**: An open source platform targeted mainly at mobile terminals and promoted by Google Inc., in the United States. Android™ is a trademark or registered trademark of Google Inc. in the United States.

[*4] **SensorManager**: A class used in Android application development. It is called when using device-mounted sensors to obtain output values and perform other sensor-related functions.

[*5] **Application framework**: An architecture for configuring Android OS consisting of component groups required for implementing standard functions in Android applications.

tion sensor[*6], for example. That is to say, we implemented the new sensor as one belonging to the same class of currently installed sensors and made it independent of existing UI functions on the layers below the application framework thereby avoiding any conflict as a UI device. This also made it possible to use the new sensor itself from the application software layer while also ensuring extendibility of the new sensor to uses other than terminal operation as in the development of novel applications using grip features.

3.3 UI Integration

As described in section 3.2, tactile-sensor information comes up through SensorManager in Android OS, so a connection between the application and SensorManager must be made to achieve a UI function.

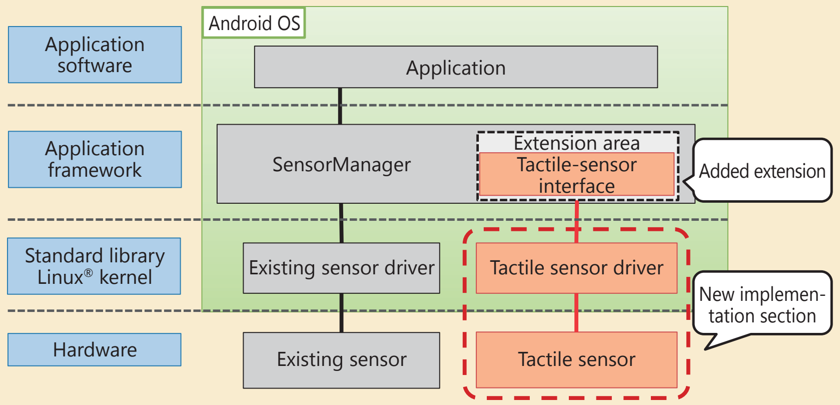
In this development, we achieve this function by incorporating a special Service[*7] between the sensor and application to perform conversion processing to a UI event provided in the Android standard. This enables input to an application based on Grip UI and input based on the existing UI to be handled as a UI event on the same level and to be used without having to modify any application.

A conceptual image of this function is shown in **Figure 5** in comparison with the existing technique. In the conventional configuration shown on the left in the figure, Event[*8]B sent from the new interface needs to be received, so a corresponding EventListener[*9]B must be added to the existing application. However, in the new development shown on the right, a newly added UI swap service determines how the user is gripping the terminal from tactile-sensor values and converts that information to a command for starting up an appropriate function. This swap service can also support UI events such as a KeyEvent reflecting the pressing of a particular key or the issuing of an intent[*10] to a particular application. These event-issuing functions can be used, for example, to start up an application or perform a simple application operation.

4. Prototype Test Results

4.1 Grip UI-equipped Mobile Terminal

An external view of our prototype terminal equipped with the Grip UI function is shown in **Figure 6**. The tac-



Linux®: A registered trademark or trademark of Linus Torvalds in the United States and other countries.

Figure 4 Platform architecture

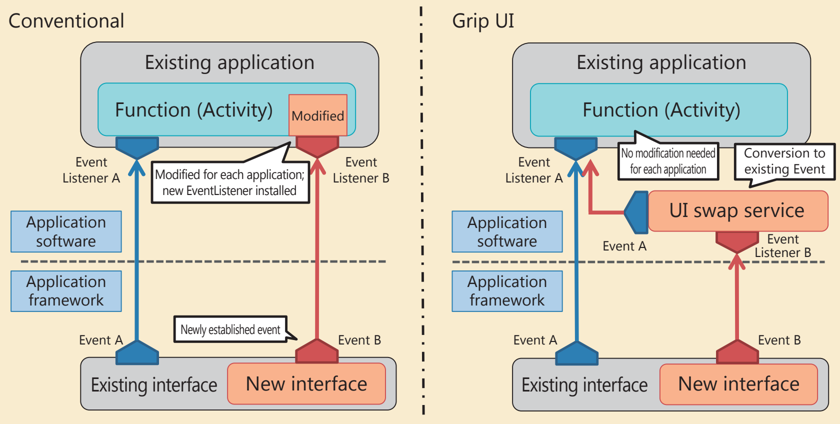


Figure 5 UI swap service

*6 **Acceleration sensor**: A sensor that measures changes in speed. Equipping a mobile terminal with an accelerometer allows it to sense orientation and motion.

*7 **Service**: An element making up an application. A function running invisibly in the background.

*8 **Event**: A notification sent to a program whenever some type of action has occurred such as a button operation.

*9 **EventListener**: A method called within a program when its corresponding event occurs.

*10 **Intent**: A mechanism provided by the Android OS for programs to exchange parameters. Used between components within an application and between applications to exchange information.

tile sensor is arranged on both sides of the terminal and on the upper part of the backside. As shown in **Table 2**, the size of this terminal is essentially the same as the base terminal (it is less than 1 mm larger), but we expect this increase in size to be sufficiently absorbed at the commercial stage.

Examples of obtaining grip features from this prototype terminal are shown in **Figure 7**. In this way, we were able to confirm that an optimal arrangement of tactile sensor elements can achieve a terminal that enables the grip features described in section 2.1 to be obtained.

4.2 Grip UI Application

To achieve a Grip UI, we carried out the implementation described in sections 3.2 and 3.3 and installed the following functions to satisfy the usability requirements listed in Table 1.

- Start-up/terminate UI swap service
- Register UI swap map
- Register grip template
- Change tactile sensor parameters

Examples of setting screens for various functions are shown in **Figure 8**. To begin with, the UI-swap-map registration function in the upper part of the figure enables the user to register and edit a correspondence between a grip-based input and a UI event that the user would like to be issued. Here, the user may simply select a previously prepared item from the displayed list to begin using the corresponding grip pattern immediately. The user may also select gripping style and issuing event as desired, that is, the user may customize the correspondence between grip-based input and a UI event. Next, the grip template registration function shown in the lower part of the figure enables the user to register and edit a grip template for the purpose of classifying a grip-based input according to a registered pattern. Here, major ways of gripping have been registered beforehand to sim-

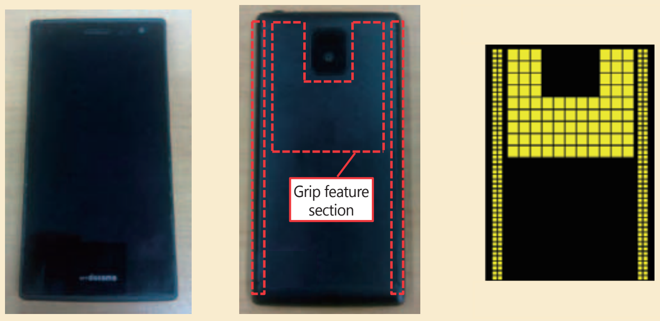


Figure 6 External view of prototype terminal

Table 2 Prototype dimensions and comparison with base terminal

	Prototype terminal	Base terminal comparison
Height	129.5 [mm]	+ 0.5 [mm]
Width	65.6 [mm]	+ 0.6 [mm]
Thickness	10.4 [mm]	± 0.0 [mm]

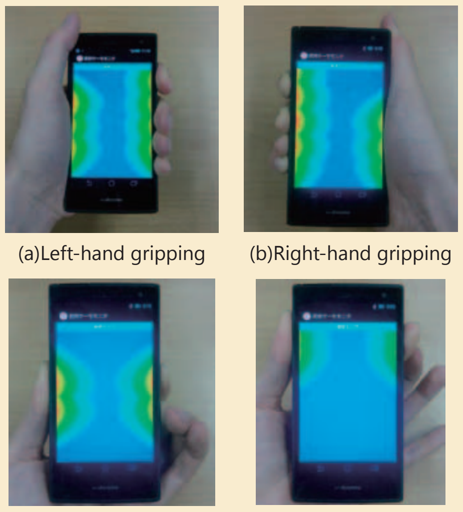


Figure 7 Examples of obtaining grip features

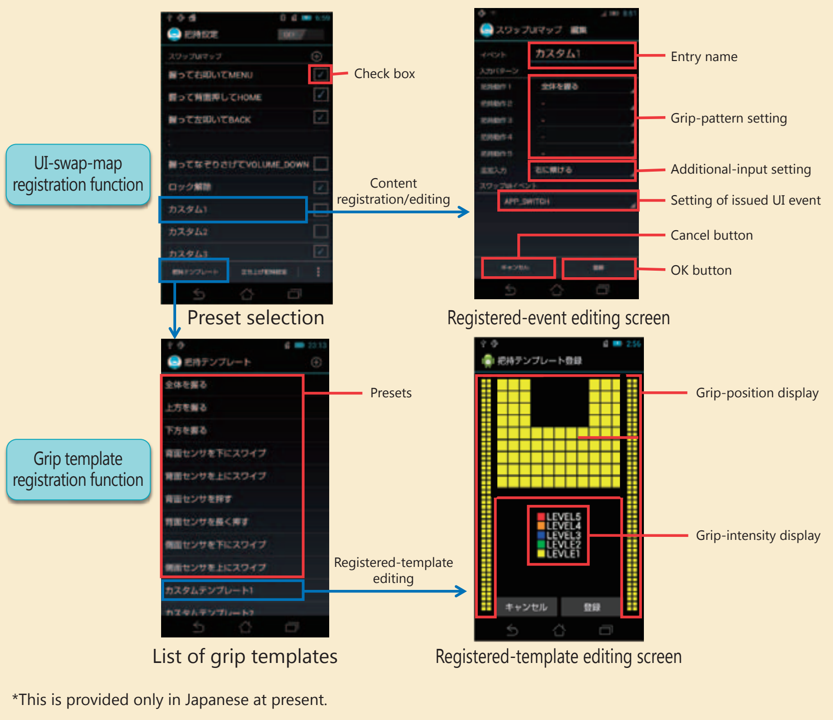

UI-swap-map registration function
Check box
Content registration/editing
Preset selection
Entry name
Grip-pattern setting
Additional-input setting
Setting of issued UI event
Cancel button
OK button
Registered-event editing screen
Grip template registration function
Presets
Registered-template editing
List of grip templates
Grip-position display
Grip-intensity display
LEVEL5
LEVEL4
LEVEL3
LEVEL2
LEVEL1
Registered-template editing screen
*This is provided only in Japanese at present.

Figure 8 Examples of Grip UI setting screens

plify the selection of gripping style at the time of map registration. The user can also specify favorites ways of gripping independently and register them while verifying grip positions and their intensity on a screen.

The above mechanism enables gripping styles and issuing events appropriate for each user to be registered and sequences of user-friendly operations to be set.

4.3 Usage Example

The user operation sequence to move from the browser application to the startup of the mail application is shown in **Figure 9** for Grip UI and the normal UI. As shown here, the mail application can be started up with a single operation by registering a gripping style and appli-

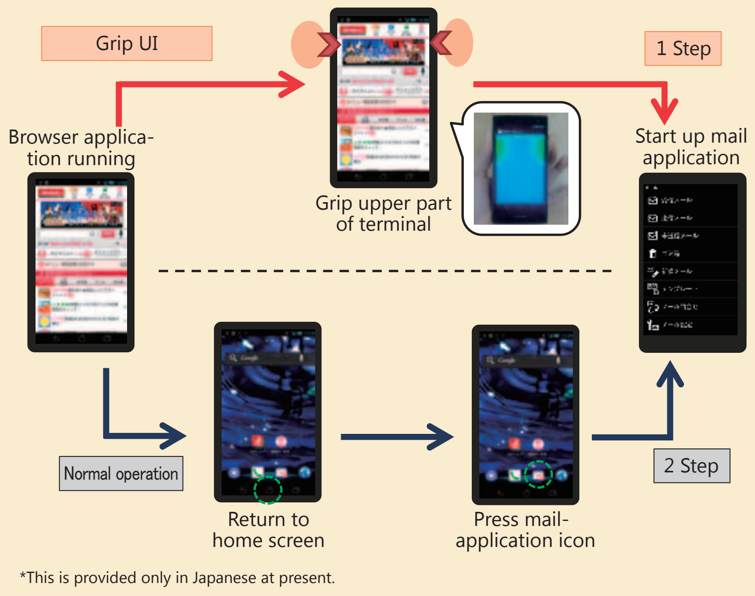

Grip UI
1 Step
Browser application running
Grip upper part of terminal
Start up mail application
Normal operation
Return to home screen
Press mail-application icon
2 Step
*This is provided only in Japanese at present.

Figure 9 Transition from browser application to mail-application startup

cation-startup event beforehand. The same result can be achieved by the conventional method of returning to the list of applications by pressing the home button and then pressing the mail-application icon. This mechanism makes terminal operations even easier to perform by registering a gripping style and corresponding event beforehand on a user-by-user basis while enabling Grip UI operations to coexist with ordinary UI operations.

5. Conclusion

We have developed a new UI focusing on the simple user action of "gripping a smartphone" and have shown with a prototype terminal that the state of gripping the terminal can be detected in detail without having to modify terminal shape. We have also shown that each user's natural gripping style can be applied to terminal operations.

We envision a variety of applications for Grip UI in addition to making one-handed operation of a smartphone more convenient as taken up in this study. For example, combining the motions of "gripping" and "tapping" should make for an even greater diversity of operations. Furthermore, as the act of gripping a terminal is considered to be an individual feature, a study has been made on applying gripping to personal authentication [6]. In short, this novel interface should be able to give added value to terminals in addition to enhancing usability.

In future research, we aim to create new usage scenarios and services by linking information on gripping style with a variety of functions.

References

[1] K.-E. Kim, W. Chang, S.-J. Cho, J. Shim, H. Lee, J. Park, Y. Lee and S. Kim: "Hand grip pattern recognition for mobile user interfaces," Proc. of the National Conference on Artificial Intelligence, Vol. 21, No. 2, pp. 1789- 1794, Jul. 2006.

[2] H. Lee, W. Chan, J. Park and J. Shim: "New mobile UI with hand-grip recognition," CHI'09 Extended Abstracts on Human Factors in Computing Systems, pp. 3521-3522, ACM, 2009.

[3] Nextinput Inc.: "ForceTouch." http://www.nextinput.com/t/ForceTouch

[4] Touchence Inc.: "ShokacCube™." http://www.touchence.jp/en/cube/index.html

[5] Nippon Mektron, Ltd.: "Flexible Tactile Sensor." http://www.mektron.co.jp/technology_e/new_tactile_sensor_fpc/

[6] T. Iso, T. Horikoshi, M. Tsukamoto and T. Higuchi: "Personal feature extraction via grip force sensors mounted on a mobile phone: authenticating the user during key-operation," Proc. of the 11th International Conference on Mobile and Ubiquitous Multimedia, ACM, 2012.