SynCro: Context-Aware User Interface System for Smartphone–Smartwatch Cross-Device Interaction

Yuki Kubo
University of Tsukuba
Tennodai 1-1-1, Tsukuba,
Ibaraki, 305-8573 Japan
kubo@iplab.cs.tsukuba.ac.jp

Buntarou Shizuki
University of Tsukuba
Tennodai 1-1-1, Tsukuba,
Ibaraki, 305-8573 Japan
shizuki@cs.tsukuba.ac.jp

Ryosuke Takada
University of Tsukuba
Tennodai 1-1-1, Tsukuba,
Ibaraki, 305-8573 Japan
rtakada@iplab.cs.tsukuba.ac.jp

Shin Takahashi
University of Tsukuba
Tennodai 1-1-1, Tsukuba,
Ibaraki, 305-8573 Japan
shin@cs.tsukuba.ac.jp

CHI'17 Extended Abstracts, May 06-11, 2017, Denver, CO, USA
ACM 978-1-4503-4656-6/17/05.
http://dx.doi.org/10.1145/3027063.3053088

Abstract
We present a context-aware user interface system, called SynCro, comprising a smartphone and a smartwatch. SynCro provides the user with context-dependent user interfaces (UIs), and will synthesize layouts, feedback, or input methods of these devices during use, depending on a identified usage context. To develop SynCro, we implemented a context recognizer that uses the smartphone and smartwatch accelerometers. The recognizer can identify 24 contexts relating to the smartphone grip, user arm posture, and user activity. Further, we implemented example applications that change the UI depending on the context identified.

Author Keywords
Multi-devices; wearable; mobile; context-aware computing; interaction design; muti-tasking.

ACM Classification Keywords
H.5.2. [Information Interfaces and Presentation]: User Interfaces—Input devices and strategies, Interaction styles.

Introduction
Cross-device interaction between devices such as tablets, smartphones, or PCs and smartphones is a popular area of research in the field of human computer interaction. This paper focuses on cross-device interaction between a

smartphone and smartwatch; this is because both devices are commonly used daily, contain several built-in sensors, and have considerable computational power.

Pioneering work that showed the potential of the combination of a smartphone/smartwatch in enhancing interactions was carried out by Duet [4]. The work explored and enhances cross-device interactions by coordinating the user interfaces (UIs) of both devices based on their spatial configurations.

However, in mobile computing environments, both devices are frequently unavailable for simultaneously use. For example, a smartphone is frequently placed into a pocket; alternatively, a user may not see a smartwatch display if they are holding a child's hand when crossing a street. Both examples suggest that the usability and design of cross-device interactions depends heavily on where both devices are placed. Thus, context-awareness in cross-device applications is important.

To further exploration of cross-device interactions between a smartphone and a smartwatch, in this study, we designed a context-aware UI system, called SynCro, comprising a smartphone and a smartwatch. To develop the system, we first implemented a context recognizer based on machine learning, using smartphone and smartwatch accelerometers; the accelerometers are used only to explore the simplest form of context recognition. We then implemented a system that uses the identified contexts to provide the user with suitable UIs; the system synthesizes layouts, feedback, or input methods for these devices. The map application shown in Figure 1 is an example.

Related Work

We reviews prior work exploring cross-device interactions, and interaction techniques that use context sensing.

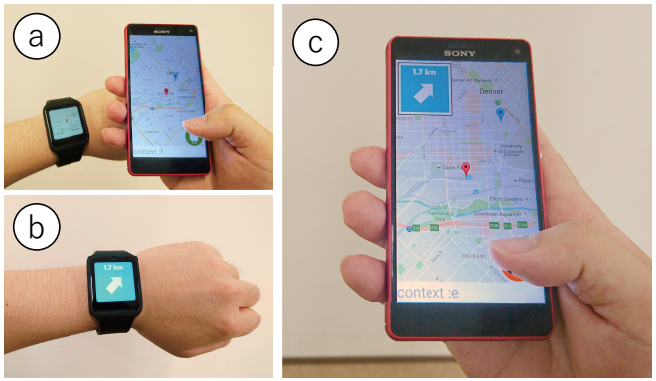

Figure 1: Map application. (a) The user can use the wide screen of the smartphone to browse the map and select a destination with their thumb. (b) After destination selection, the smartwatch displays the distance and direction with an arrow, allowing the user to see the information easily while walking. (c) The smartwatch screen is mirrored on the smartphone when the user lowers their left arm. The user can now zoom in or out using a wrist tilt gesture of their left arm; this allow the user to browse the map easily with the right hand to check whether they are on the correct route.

Cross-Device Interactions

Several studies have explored cross-device interaction between two or more devices. For example, Pick-and-drop [11] is a method with which the user can use a pen to transfer data between multiple displays. Yoon et al. [15] proposed a cross-device interaction that combines grasp and micro-mobility on a tablet. Schmidt et al. [13] proposed a cross-device interaction style that utilizes a smartphone as a tangible input object to a large display. Hinckley et al. [7] explored a technique that combine grip and motion sensing on a pen and tablet. In contrast to the above, our cross-device interaction uses a smartphone and

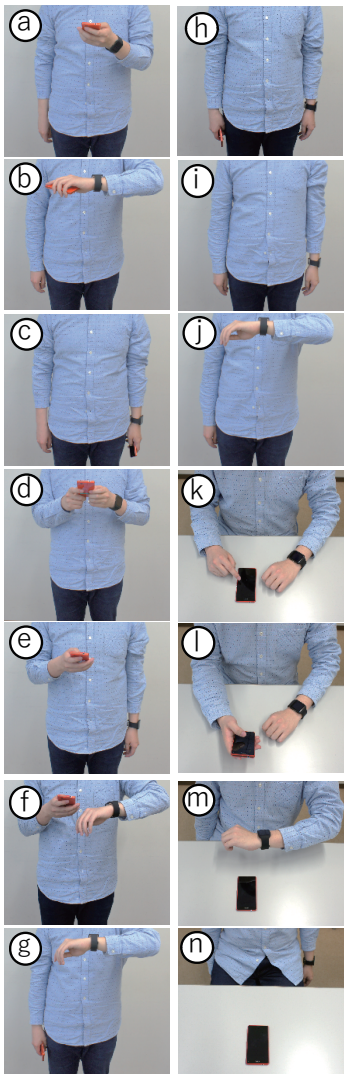

Figure 2: Contexts. a–j can occur while walking.

smartwatch, and Duet [4]— the work most inspired us— uses the same device. For example, the smartwatch may be used as a tool palette; it could alternatively serve as a sub-display, displaying contents of a clipboard on the smartphone; as a third option, a map on the smartphone could be zoomed-out by bumping the smartphone twice on the smartwatch. TakeOut [10] proposed a drawing application using a smartphone and a smartwatch, utilizing them as a canvas and palette respectively. In our study, we explore cross-device interactions between a smartphone and a smartwatch based on context-awareness.

Interaction Techniques using Context Sensing

Some studies have shown the potential of contextual sensing in improving interactions. Schilit et al. [12] showed that it is possible to provide an optimal UI based on a given situation. For example, Proximate Selection is a UI technique that makes a choice easy based on user's location information. Hinckley et al. [8] proposed context-aware mobile interactions using several sensors; for example, their system rotates the UI to adjust to the device's orientation. Yang et al. [14] changed the application running on a smartwatch based on the hand posture, which was recognized by electromyographic sensors attached to the arm. iGrasp [5] changes the keyboard layout based on grip recognition by the device case, which has embedded capacitive touch sensors. Mo-Bi [9] uses bimanual hand postures, recognized using the accelerometers of one smartphone and a wrist-worn devices on each hand, to interface layouts and functions in posture-related applications. By contrast, our work provides UIs of both the smartphone and the smartwatch, and which are suitable for contexts in a smartphone-smartwatch cross-device interaction.

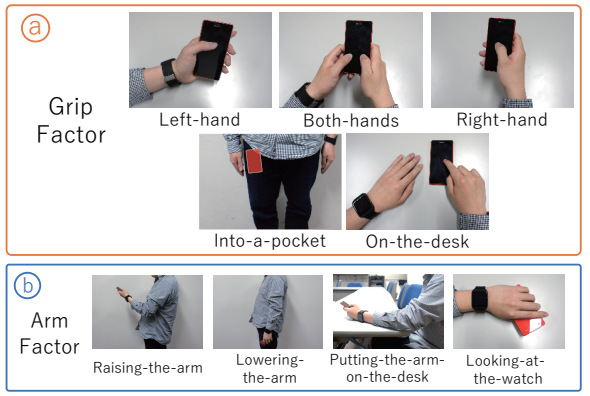

Figure 3: Context factors.

Contexts

Our context-aware UI system provides the user with both smartphone and smartwatch UIs. In this section, we described the types of contexts that our recognizer can identify, and the UIs for cross-device interactions, based on the identified contexts.

The contexts that our recognizer can identify are shown in Figure 2, with the assumption that the user wears a smartwatch on the left wrist. Among these contexts, a–j can occur while walking, and have therefore been identified as separate contexts a'–j'; the k–n contexts can only occur while resting in a chair.

The contexts are determined by three factors: How a smartphone is gripped (*grip*), the user's arm posture (*arm*), and the user's activity (*activity*). The grip factor has five levels, as shown in Figure 3a, and adds three further levels: Left-hand, right-hand, and both-hands. It also has two additional levels: Into-a-pocket and on-the-desk. We included these two levels into the grip factor for the

Arm Factors \ Grip Factors	Left hand	Both hands	Right hand: Raising the arm	Right hand: Lowering the arm	In the Pocket	On the desk
Raising the arm	a	d	f	g	j	m
Lowering the arm	c	N / A	e	h	i	n
the arm on the desk	a	N / A	l	g	--	k
Looking at the watch	b	N / A	f	g	j	m

Table 1: Relations between the grip and arm factors and the contexts, when resting.

sake of convenience, but the smartphone is not gripped in the user's hand. The arm factor has four levels as shown in Figure 3b. The activity factor has two levels: resting or walking.

Relations between the grip and arm factors and the identified contexts utilizing the two devices are shown in Tables 1 and 2. Note that our recognizer can identify the arm factor of the right hand only when the smartphone is gripped in that hand. In these tables, N/A denotes the contexts that do not exist; these include under both-hands, lowering-the-arm, putting-the-arm-on-the-desk, and looking-at-the-watch. We have also merged similar contexts into one; this is the reason one ID appears two or more times in the tables. Specifically, we merged all "raising-the-arm of the left-hand" with "looking-at-the-watch" if the smartphone is not gripped in the left arm; this is because raising the left arm is a preliminary movement before looking-at-the-watch. Furthermore, "–" in Table 1 denotes the context that our recognizer does not try to identify; we incorrectly thought that putting a smartphone into-a-pocket could occur only when standing.

Arm Factors \ Grip Factors	Left hand	Both hands	Right hand: Raising the arm	Right hand: Lowering the arm	In the Pocket
Raising the arm	a'	d'	f'	g'	j'
Lowering the arm	c'	N / A	e'	h'	i'
Looking at the watch	b'	N / A	f'	g'	j'

Table 2: Relations between the grip and arm factors and the contexts, when walking.

Implementation

To realize our context-aware UI system, we implemented a context recognizer based on machine learning using the accelerometers in a smartphone and smartwatch.

Devices and Configuration

We used a SONY Xperia Z3 Compact SO-02G, and a SO-NY SmartWatch 3 SWR50. The smartphone has a quad-core 2.5 GHz processor and 2 GB random access memory (RAM). The smartwatch has a quad–core 1.2 GHz processor and 512 MB of RAM. The smartwatch was connected to the smartphone via Bluetooth, and the transmission rate of the sensor data was set to 20 Hz. We empirically determined that this frequency achieves a stable transmission to the smartphone.

Context Recognizer

Our context recognizer uses the J48 decision tree classifier in the WEKA data mining software [6], with the default setting (confidence factor = 0.25) to identify contexts. It runs on the smartphone and collects the 3-axis acceleration values from the smartphone ($a_{px}(t)$, $a_{py}(t)$, $a_{pz}(t)$) and smartwatch ($a_{wx}(t)$, $a_{wy}(t)$, $a_{wz}(t)$).

To identify the grip and arm factors, the recognizer calculates the sum of squares of acceleration $a_p(t)^2$, $a_w(t)^2$ from each device. The recognizer also calculates $a_d(t)^2$, which is the subtraction of $a_w(t)^2$ from $a_p(t)^2$. Details of these values are:

$$\begin{aligned} a_p(t)^2 &= a_{px}(t)^2 + a_{py}(t)^2 + a_{pz}(t)^2 \\ a_w(t)^2 &= a_{wx}(t)^2 + a_{wy}(t)^2 + a_{wz}(t)^2 \\ a_d(t)^2 &= a_w(t)^2 - a_p(t)^2 \end{aligned}$$

In addition, to identify activity (i.e., resting or walking), we used a fast Fourier transform (FFT) to calculate the following features:

Frequency Power [3]: We used FFT to each $a_p(t)^2$, $a_w(t)^2$, and $a_d(t)^2$, with a sliding window of 32 samples. The frequency ranges is therefore 0–5 Hz with the resolution of 0.31 Hz; this is because the sliding window is 32 and the sampling rate is 20 Hz. This window produces 16 frequency powers.

Maximum Frequency Power: The maximum frequency power in FFT for each $a_p(t)^2$, $a_w(t)^2$, and $a_d(t)^2$.

Frequency of Maximum Frequency Power: The frequency that shows the Maximum Frequency Power for each $a_p(t)^2$, $a_w(t)^2$, and $a_d(t)^2$.

Average Acceleration [2]: The average acceleration in a sliding window for each axis of both the smartphone and smartwatch.

Average Difference Acceleration: The average of $a_d(t)^2$ in a sliding window.

Average Resultant Acceleration [1]: The average of each $a_p(t)^2$ and $a_w(t)^2$ in a sliding window.

The total number of features is 63: 48 Frequency Powers, three Maximum Frequency Powers, three Frequencies of Maximum Frequency Power, six Average Accelerations, one Average Difference Acceleration, and two Average Resultant Accelerations.

Example Applications

We implemented the following applications to demonstrate context-aware UI systems comprising a smartphone and smartwatch.

Map

In the smartphone–smartwatch cross-device interaction described in [4], a smartwatch serves as a sub-display showing the map application. We extended this application, to change UIs automatically to make it suitable for different contexts (Figure 1). Figure 1a is the map UI when a user raises both hands; in this UI, the user can use the wide screen of the smartphone to browse the map and select a destination with their thumb. After destination selection, the smartwatch displays the distance and direction with an arrow (Figure 1b), allowing the user to see the information easily while walking. The smartwatch screen is mirrored on the smartphone when the user lowers the left arm (Figure 1c). Using this UI, the user can now zoom in or out using a wrist tilt gesture of their left arm; this allows the user to browse the map easily with the right hand to check whether they are on the correct route.

Notification Management System

We also implemented a smartphone–smartwatch notification management system, which changes notified devices

Figure 4: Multi-tasking application. (a) The user can transmit the current application by pushing the button. The application is then transmitted to the smartwatch and can be operated. (b) The user can use both applications by using two devices simultaneously. (c) While lowering the left arm, the display of the smartwatch is mirrored onto the smartphone, which allows the user to continue to perform multi-task, even while lowering the left arm.

Contexts	Notified device and vibration
a, d, e	smartphone, non-vibration
b, g, j, m	smartwatch, non-vibration
c, h	smartphone, vibration
f, i, k, l, n, a'–j'	smartwatch, vibration

Table 3: Notifications. In this table, non-vibration denotes that the system only displays the notification on the device's display, and the device does not vibrate. In contrast, vibration denotes that the system displays the notification on the device and vibrates it.

and notification methods (display or vibration) depending on contexts; this is shown in Table 3.

This system displays notifications on the smartphone without any vibration when the user operates the smartphone (a, d, e). Similarly, it displays notifications on the smartwatch without any vibration when the user operates the smartwatch (b, g, j, m). In contrast, the system displays notifications on, and vibrates the smartphone when the user does not operate the device, but holds it (c, h). Similarly, the system displays notifications on the smartwatch when the user neither holds the smartphone (i, n) nor operates the two devices (f, k, l). Moreover, the system displays all notifications on, and vibrates the smartwatch, to prevent manipulation while walking (a'–j').

Multi-tasking Application

A multi-tasking application was also implemented; the use case shown is where the user can use a music player and web browser simultaneously (Figure 4). This application displays a button on the current application; when the user pushes the button, the current application in this case the music player is mirrored to the smartwatch and can be operated. As shown in Figure 4b, this allows another application, for example browser, to be used on the smartphone. The application also mirrors the smartwatch screen on the smartphone when the user lowers the left arm (Figure 4c). In the smartwatch's screen on a smartphone, the user can operate the music player with the his/her finger, and therefore, can continue to perform multi-tasking even while lowering the left arm.

Conclusions

We presented a context-aware UI system, called SynCro, comprising a smartphone and a smartwatch. SynCro provides the user with UIs suitable for contexts; it synthesizes layouts, feedback, or input methods of these devices depending on the context identified. To develop this system, we first implemented a context recognizer that uses the smartphone and smartwatch accelerometers. This recognizer can identify 24 contexts relating to how the smartphone is gripped, the user's arm posture, and the user's activity. Three example applications have demonstrated the usefulness of context-aware UI systems consisting of a smartphone and smartwatch; the application types are a map, a notification management system, and a multi-tasking application.

In the future, we plan to conduct an experiment to evaluate the accuracy of the context recognizer in controlled and non-controlled conditions and the usability of the example applications. Further, we will explore how accuracy improves when we add sensors such as gyroscopes and magnetometers. Furthermore, we plan to conduct an experiment to evaluate the accuracy of the context recognizer based on other machine learning methods.

Acknowledgements

This work was partially supported by ACT-I, Japan Science and Technology Corporation (JST).

References

[1] Ian Anderson, Julie Maitland, Scott Sherwood, Louise Barkhuus, Matthew Chalmers, Malcolm Hall, Barry Brown, and Henk Muller. 2007. Shakra: Tracking and Sharing Daily Activity Levels with Unaugmented Mobile Phones. *Mobile Networks and Applications* 12, 2 (2007), 185–199. DOI: http://dx.doi.org/10.1007/s11036-007-0011-7

[2] Ling Bao and Stephen S. Intille. 2004. Activity Recognition from User-Annotated Acceleration Data. In *Proceedings of the 2nd International Conference on Pervasive Computing (Pervasive' 04)*. Springer, 1–17. DOI: http://dx.doi.org/10.1007/978-3-540-24646-6_1

[3] Agata Brajdic and Robert Harle. 2013. Walk Detection and Step Counting on Unconstrained Smartphones. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '13)*. ACM, 225–234. DOI: http://dx.doi.org/10.1145/2493432.2493449

[4] Xiang 'Anthony' Chen, Tovi Grossman, Daniel J. Wigdor, and George Fitzmaurice. 2014. Duet: Exploring Joint Interactions on a Smart Phone and a Smart Watch. In *Proceedings of the 32nd SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, 159–168. DOI: http://dx.doi.org/10.1145/2556288.2556955

[5] Lung-Pan Cheng, Hsiang-Sheng Liang, Che-Yang Wu, and Mike Y. Chen. 2013. iGrasp: Grasp-based Adaptive Keyboard for Mobile Devices. In *Proceedings of the 31st SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, 3037–3046. DOI: http://dx.doi.org/10.1145/2470654.2481422

[6] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations Newsletter* 11, 1 (Nov. 2009), 10–18. DOI: http://dx.doi.org/10.1145/1656274.1656278

[7] Ken Hinckley, Michel Pahud, Hrvoje Benko, Pourang Irani, François Guimbretière, Marcel Gavriliu, Xiang 'Anthony' Chen, Fabrice Matulic, William Buxton, and Andrew Wilson. 2014. Sensing Techniques for Tablet+Stylus Interaction. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, 605–614. DOI:http://dx.doi.org/10.1145/2642918.2647379

[8] Ken Hinckley, Jeff Pierce, Mike Sinclair, and Eric Horvitz. 2000. Sensing Techniques for Mobile Interaction. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology (UIST '00)*. ACM, 91–100. DOI: http://dx.doi.org/10.1145/354401.354417

[9] Han-Jong Kim, Seijin Cha, Richard C. Park, Tek-Jin Nam, Woohun Lee, and Geehyuk Lee. 2016. Mo-Bi: Contextual Mobile Interfaces through Bimanual Posture Sensing with Wrist-Worn Devices. In *Proceedings of HCI Korea (HCIK '16)*. Hanbit Media, Inc., 94–99. DOI: http://dx.doi.org/10.17210/hcik.2016.01.94

[10] Woori Noh, Mankyung Lee, Hyelim Cheon, Joohee Kim, Kwangjae Lee, and Jundong Cho. 2016. Take-Out: Drawing Application using Distributed User Interface for Being Close to Real Experience. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct (UbiComp '16)*. ACM, 173–176. DOI: http://dx.doi.org/10.1145/2968219.2971439

[11] Jun Rekimoto. 1997. Pick-and-drop: A Direct Manipulation Technique for Multiple Computer Environments. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology (UIST '97)*. ACM, New York, NY, USA, 31–39. DOI: http://dx.doi.org/10.1145/263407.263505

[12] Bill N. Schilit, Norman Adams, and Roy Want. 1994. Context-Aware Computing Applications. In *Proceedings of the 1st Workshop on Mobile Computing Systems and Applications (WMCSA '94)*. IEEE Computer Society, 85–90. DOI: http://dx.doi.org/10.1109/WMCSA.1994.16

[13] Dominik Schmidt, Julian Seifert, Enrico Rukzio, and Hans Gellersen. 2012. A Cross-device Interaction Style for Mobiles and Surfaces. In *Proceedings of the Designing Interactive Systems Conference (DIS '12)*. ACM, 318–327. DOI: http://dx.doi.org/10.1145/2317956.2318005

[14] Yoonsik Yang, Seungho Chae, Jinwook Shim, and Tack-Don Han. 2015. EMG Sensor-based Two-Hand Smart Watch Interaction. In *Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology (UIST '15 Adjunct)*. ACM, 73–74. DOI: http://dx.doi.org/10.1145/2815585.2815724

[15] Dongwook Yoon, Ken Hinckley, Hrvoje Benko, François Guimbretière, Pourang Irani, Michel Pahud, and Marcel Gavriliu. 2015. Sensing Tablet Grasp + Micro-mobility for Active Reading. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology (UIST '15)*. ACM, 477–487. DOI: http://dx.doi.org/10.1145/2807442.2807510