Journal of Computer Science 10 (7): 1238-1248, 2014
ISSN: 1549-3636
© 2014 Science Publications
doi:10.3844/jcssp.2014.1238.1248 Published Online 10 (7) 2014 (http://www.thescipub.com/jcs.toc)

ADAPTIVE I/O SWITCHING FOR SEAMLESS CONVERGENCE OF SMART PHONES AND EXTERNAL I/O DEVICES

Oh-Chul Kwon, Jusung Kim and Chang-Gun Lee

Department of Computer Science and Engineering, Seoul National University, Seoul, South Korea

Received 2014-01-15; Revised 2014-02-17; Accepted 2014-02-21

ABSTRACT

To be small and easy to carry, mobile devices cannot be equipped with large and highly capable input/output devices. In order to overcome these limitations, this study proposes adaptive I/O switching technique as a general solution for exploiting full features of various external input/output devices in lieu of tiny input/output devices in a smartphone. Unlike existing physical-layer connection based solutions or application-layer non-system solutions, we propose a generalized solution at the mobile platform level. With this generalized solution, whatever activities of a smartphone can be resolution-adaptively displayed in an external display device. Also, the generalized solution allows us to use any kinds of external input devices for general mobile applications even if they are not designed so. Adaptive I/O Switching technique is actually implemented on open source mobile platform and its effectiveness is demonstrated with several example scenarios.

Keywords: Adaptive I/O Switching, External I/O Device, Mobile Platform, Smartphone

1. INTRODUCTION

Mobile devices like cell phones and tablets set us free from locational restrictions. Moreover, thanks to increasing processor speeds and radio communication bandwidths, it becomes real for emerging smartphones to provide general computing and IT services including web browsing, video streaming and even document editing. However, the Quality of Experiences (QoEs) of those services are still very limited ironically because of the most appealing aspect of mobile devices, that is, "small and thus easy to carry". To be small and easy to carry, mobile devices cannot be equipped with large and highly capable input/output devices.

Under this inherent limitation, we need to exploit surrounding gadgets and appliances as alternative input/output devices aiming at high QoEs of general computing and IT services with smartphones. For this, solutions are possible at different levels. First, there can be a physical-layer connection based solution such as TV-out connection, WHDI (Lawton, 2008), WiDi (2010), WirelessHD (2010), Miracast (2013) and Bluetooth (2013). Only with this layer solution, however, the external display device simply shows the magnified but the same image of the smartphone. That is, it is not possible to reconstruct the images exploiting the full resolution of the external display device. In the input side, only the physically compatible input devices can be used for mobile applications. Second, there can be an application-layer non-system solution. However, it can transfer images to external display devices only for a specifically targeted mobile application but not for general activities of the smartphone. Also in the input side, it can allow us to use only a specifically targeted external input device for a specifically targeted mobile application but not for general input devices and general mobile applications.

In order to overcome these limitations, we propose a solution in the platform level, that is, an adaptive I/O switching technique, to achieve the following two goals:

- Output side: All the images constructed from any activities of the smartphone should be resolution-

Corresponding Author: Oh-Chul Kwon, Department of Computer Science and Engineering, Seoul National University, Seoul, South Korea Tel: +82-2-880-2562

adaptively reconstructed and displayed in the external display device

- Input side: Any kinds of external inputs should be transformed to internal input events so that they can be used to control any mobile applications even if they are originally designed without consciousness of such external inputs

In this study, for this generic solution of adaptive convergence of a smartphone and external input/output devices, we discover the limitation of the current structure of mobile platform and propose its modifications for adaptive I/O switching. The modified structure are actually implemented and demonstrated with several example scenarios.

Android (2013) is the most widely used mobile platform. Especially android is open source so that we can easily analyze the internal architecture and apply our propose technique. For these reasons, we choose Android as the target of analysis and implementation in this study.

The rest of this study is organized as follows: The next section briefly surveys related work. Section 3 and section 4 address the adaptive I/O switching each. Section 5 addresses our implementation. In Section 6, evaluation and comparison are discussed. Finally, section 7 concludes this study.

2. RELATED WORK

To address the tiny display problem of mobile devices, there are many techniques for using external display devices. The latest smartphones with Mobile High-Definition Link (MHL, 2008) can connect them to a TV to watch videos and photos via High Definition Multimedia Interface (HDMI, 2009). Greaves *et al*. (2008) uses a different approach using a projector to display the mobile device's screen and Singh *et al*. (2010) create a low cost interactive electronic whiteboard by using a projector and Nintendo Wiimotes. Also, there are several WirelessHD protocols such as WHDI, WiDi, WirelessHD and Miracastto beam a display wirelessly from a small device to TV or projector.

These techniques, however, provide only physical connections to external devices so that the mobile device's display can show up in the external display device, i.e., simple magnified view of the same image content of the mobile display. Thus, only with this physical-layer connection based solution, it is not possible to resolution-adaptively reconstruct the image content of the smartphone and hence we cannot exploit the full features of the external display device such as the high resolution feature of an external HDTV. Motorola Atrix (2011) exploits the external device's full feature by using a docking device called "WebDock". However, it is not a general solution since it works only with dedicateddevices made by Motorola. The I/O virtualization technique (Ha *et al*., 2010) provides a notion of a virtual driver to exploit general output devices. However, it is still in a conceptual level without any specific methodologies. In contrast to these existing techniques, our proposed solution is made in the mobile platform and Linux kernel level so that all the images from any activities of the smartphone can be resolution-adaptively reconstructed and displayed in the external display device.

The idea of resolution adaptive display switching is already available in remote desktop mechanisms such as in RDP (2013), VNC (Richardson *et al*., 1998) Citrix (2009) and IKVM (2010). However, the current mobile platform does not have such a mechanism. More seriously, it does not have structural room to embed such a mechanism. In this sense, our proposed resolution adaptive display switching technique can be understood as the first working system in the mobile platform.

Since ORiordan *et al*. (2005) investigated text input method for mobile phones, in order to overcome the limitations of mobile input devices, there are many techniques for using external input devices. For example, iControlPad (2011) and Zeemote (2012) present joypads for a mobile device using Bluetooth. Mora and Papp (2012) is a mobile application, which can accept inputs from keyboard and mouse of PC. These techniques, however, are supported only in specific applications or devices. In contrast, our technique transforms any kinds of external input data to internal input events so that they can be taken by any existing mobile applications that are originally designed without consciousness of such external inputs.

The remote desktop mechanisms such as RDP, VNC, Citrix and IKVM also allow us to use physically separated input devices, e.g., the remote keyboard/mouse. However, those mechanisms can transfer inputs only from known devices. Utilizing even unknown input devices as in our technique is not possible.

Mobizen (2012), MyMobiler (2012) and TeamViewer (2013) allow us to remotely control the smartphone by the computer. These solutions are possible to support advanced I/O switching, but only support display mirroring and restricted input devices.

3. ADAPTIVE SWITCHING OF DISPLAY DEVICES

This section addresses the issue of adaptively switching display devices. In section 3.1, we first briefly explain the drawing mechanism of Android and then discover its limitation for adaptive display switching. Section 3.2 presents our proposed adaptive display switching technique. Section 3.3 shows several use cases made possible by the proposed solution.

3.1. Drawing Mechanism of Android

Figure 1 shows the part of Android related to the display process from applications to the LCD panel of a smartphone. As shown in the figure, each mobile application has a number of image draw spaces called "surfaces". When applications draw something on their surfaces, all of the surfaces are composed by a layout module called "SurfaceFlinger", which in turn calls OpenGL (2013) functions. Finally, OpenGL draws the final display image to the framebuffer, whose content is eventually displayed on the LCD panel. For this display process to work, in the boot-up time of an Android device, the Surface Flinger gets the display information such as LCD resolution, bpp format and fps from the kernel device driver and allocates the framebuffer memory accordingly. After such boot-up time initialization of display information and framebuffer, they cannot be changed. Thus, the applications and OpenGL can draw display images only with the original display configuration.

This boot-up time configuration of display information and framebuffer is the major limitation for the current mobile platform to support the adaptive display switching.

3.2. Proposed Adaptive Display Switching

In order to overcome the aforementioned limitation, we propose the adaptive display switching technique that allows run-time reconfiguration of display information and framebuffer. **Fig. 2** shows the modified drawing mechanism for adaptive display switching. In order to dynamically reconfigure display information and framebuffer, "DisplaySwitchFunction"is implemented in SurfaceFlinger. This functionmodule is called whenever a new display device is detected (e.g., the HDMI cable is connected or UPnP (2010) found a new display device via Wi-Fi and so on) and chosen by a user. Once called, the function module performs the followings:

- Read the new display device profile
- Change the display information (device resolution, bpp, format, fps,) in layout module (SurfaceFlinger)
- Allocate a new framebuffer according to the new display information and
- Re-initialize graphic engine Open, 2013 such that it can draw the display image on the new framebuffer according to the new display information

For the new framebuffer's content to be displayed in the external display device, we also implement the "Transmitter" module, which continuously deliver the framebuffer content to the external display device through the corresponding device drivers such as Wi-Fi, HDMI and USB connection. With this modified structure, we can have more than one framebuffer at a time. In order to effectively utilize those multiple framebuffers, we also implement a multiple framebuffer management mechanism. By this mechanism, the multiple framebuffers can be concurrently used in many useful ways as will be discussed in section 3.3.

3.3. Use Cases of the Adaptive Display Switching

With modified structure, we can naturally have more than one framebuffer, that is, the original framebuffer for the smartphone's LCD panel and newly allocated framebuffers for external display devices. These multiple framebuffers make possible many handy use cases. This section presents three example use cases. The first two are already implemented and ready for demonstration and the third will be implemented in our future work.

The first use case is for the movie player application. After switching the movie screen to an external display device, we use the original LCD panel of the smartphone as a remote controller as shown in **Fig. 3**. For this, we provide the original framebuffer's address to application developers. Then, the application developer can make a movie player with two modes, normal mode and external screen mode. In the normal mode, the control buttons and scroll bars show up on the smartphone's original LCD panel overlaid over the movie screen. In the external screen mode, the movie screen is show up in the external display device with the adapted resolution while the original framebuffer for the LCD panel contains the remote controller image with buttons of play forward, play backward, pause, stop. Thus, the smartphone can now be used as a remote controller for watching the movie with the external display device.

The second use case is to use the original framebuffer as a viewfinder of a large external display device as shown in **Fig. 4**. The external display device is usually larger than the smartphone's original LCD panel.

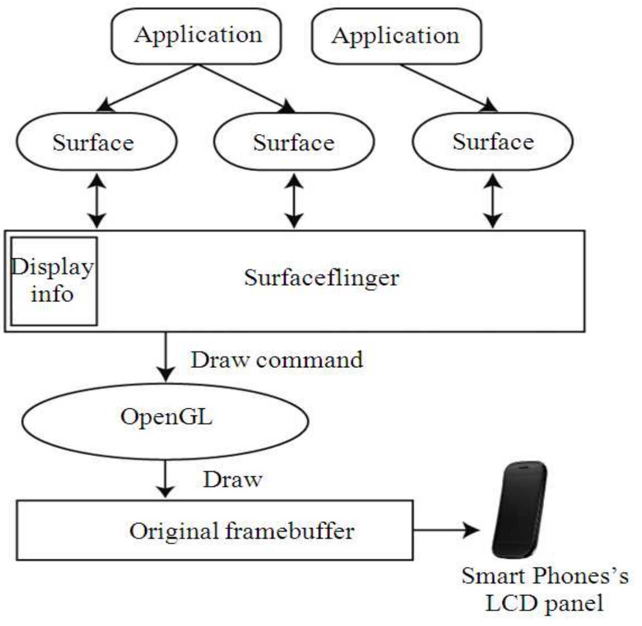

Fig. 1. Drawing mechanism of android

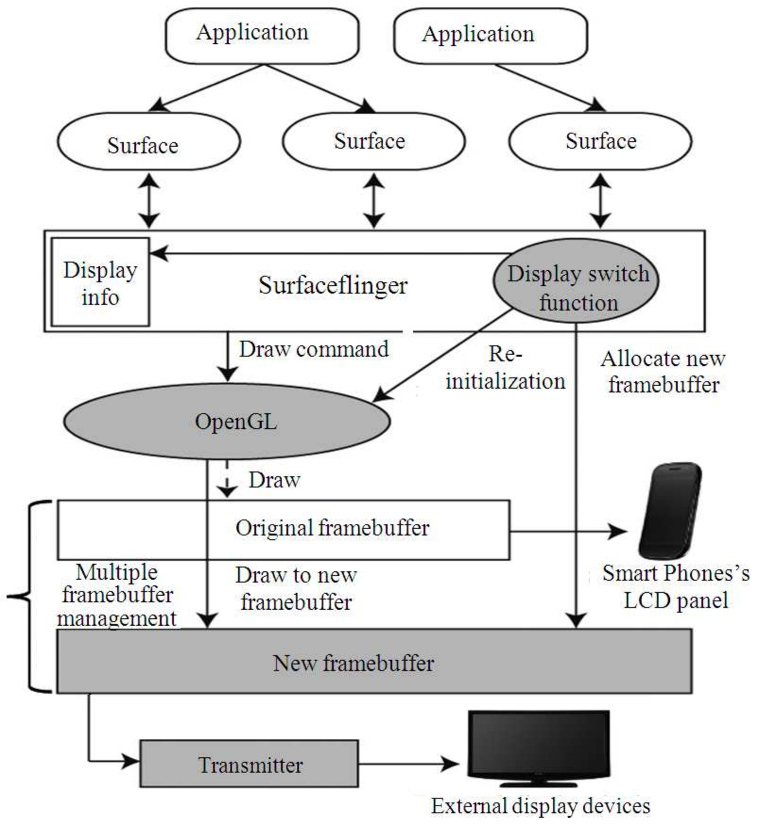

Fig. 2. Modified drawing mechanism for proposed adaptive display switching

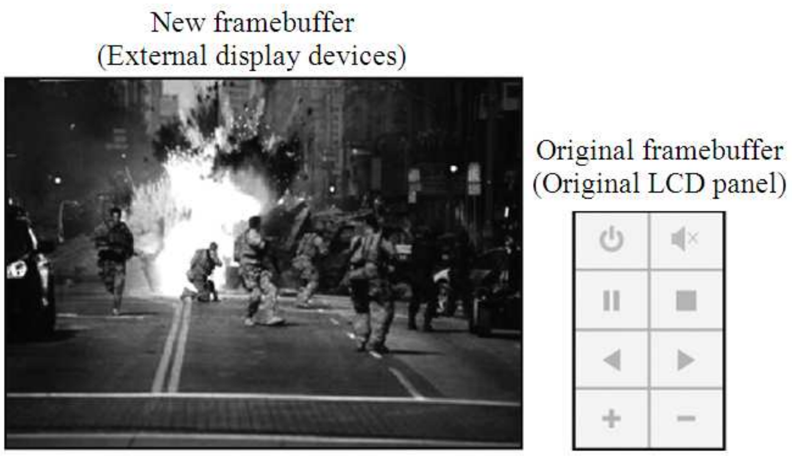

Fig. 3. Remote controller use case

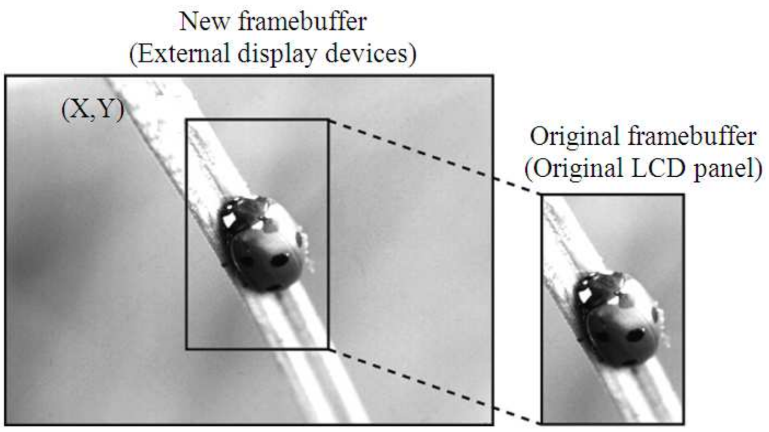

Fig. 4. Viewfinder use case

Accordingly, the resolution of the new framebuffer for the external display device is bigger than the original framebuffer. Thus, the original framebuffer cannot show the whole display image without degrading the image resolution. Therefore, in the viewfinder use case, the original framebuffer show only a small part of the new framebuffer starting from (x, y) position as in **Fig. 4**. When a user drags the smartphone's LCD panel, the starting point of the viewfinder, i.e., (x, y) is changed. So, visible part of the whole image is also changed. This viewfinder use case is effective when browsing webs or controlling the focus area of a large picture.

The third use case is for general usage of multiple framebuffers. In the current smartphone with only one fixed framebuffer, there can be only one foreground application that shows its image on the LCD panel. In contrast, proposed solution supports multiple framebuffers. Thus, it is possible to use them for multiple display devices at the same time. This implies that we can arbitrarily map multiple applications to multiple external display devices, which allows us to have multiple foreground applications at the same time. This use case will be implemented in our future work.

4. ADAPTIVE SWITCHING OF INPUT DEVICES

This section addresses the issue of adaptively switching input devices. Unlike the existing ad hoc solutions that work only with specific predefined input devices, we aim at a general solution that can take inputs from various external input devices in a unified way. In Section 4.1, we first explain the input delivery mechanism of the current mobile platform and then discover its limitation for adaptive input switching. Section 4.2 presents our proposed adaptive input switching technique. Section 4.3 shows several use cases made possible by the proposed solution.

4.1. Input Delivery Mechanism of Android

Figure 5 shows the part of the mobile platform related to the input process from input devices to applications. As shown in the figure, mobile platform generally categorizes input devices into two groups: (1) regular input devices such as QWERTY keyboard touch screen and track ball and (2) sensor input devices such as accelerometer sensor, orientation sensor and magnetic field sensor. Inputs from these two groups are handled quite differently.

First, the regular input events are handled by a permanent daemon called "InputManager" since regular inputs are always needed regardless what applications are foreground one. InputManager has two threads- "InputReaderThread" and "InputDispatcherThread". InputReaderThread continuously polls input events from regular input drivers and delivers input events if any to InputDispatcherThread. Then, InputDispatcher-Thread dispatches the input events to the current foreground application that can be found by "Window-ManagerService".

On the other hand, the sensor inputs are handled only when an executing application needs them. More specifically, when the application needs sensor inputs, it creates "SensorManager" in its "Context". At this moment, SensorManager is initialized and starts "SensorManagerThread". Then, SensorManagerThread polls sensor data from sensor drivers and notifies the changed sensor data to the application.

Although well structured, the above current input delivery mechanism works only for the predefined input devices that are embeddable in a mobile device. Thus, it is limited in adaptively using external input devices which are not originally intended to be used by applications. For example, if a racing game application is implemented to use the smartphone's touch inputs and embedded accelerometer inputs, with the current mobile platform, it is not possible to play the game with a fancier external input device with a steering wheel and foot pedals.

In order to overcome the aforementioned limitation, we propose the adaptive input switching technique as in **Fig. 6** that can transform external input data to internal input events such that the external input data can be treated as inputs from any of existing internal input devices.

In the modified mechanism, "ExternalInputDevice-Manager" communicates with external input devices through Wi-Fi or Bluetooth. For external input devices that have only wired connections such as USB or serial, we assume they are connected through a small dongle called "Input Device Hub" that converts from wired interface such as serial and USB to wireless interface such as Wi-Fi and Bluetooth as shown in **Fig. 6**.

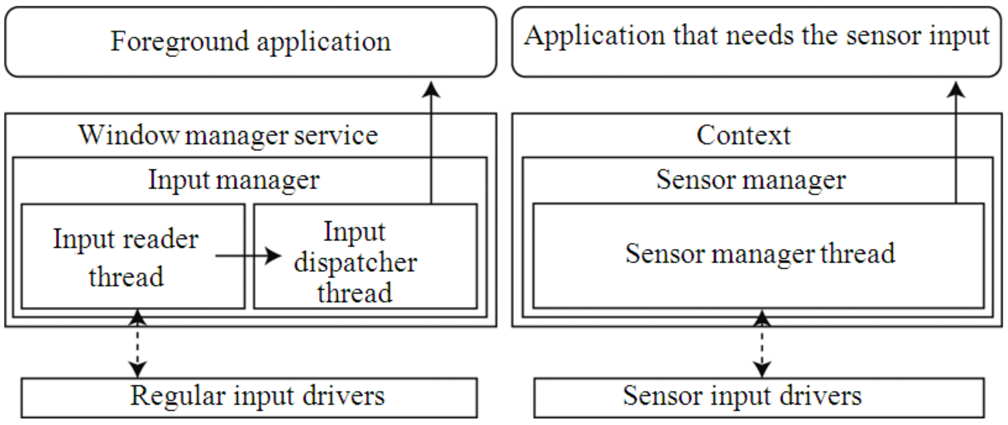

Fig. 5. Input delivery mechanism of android

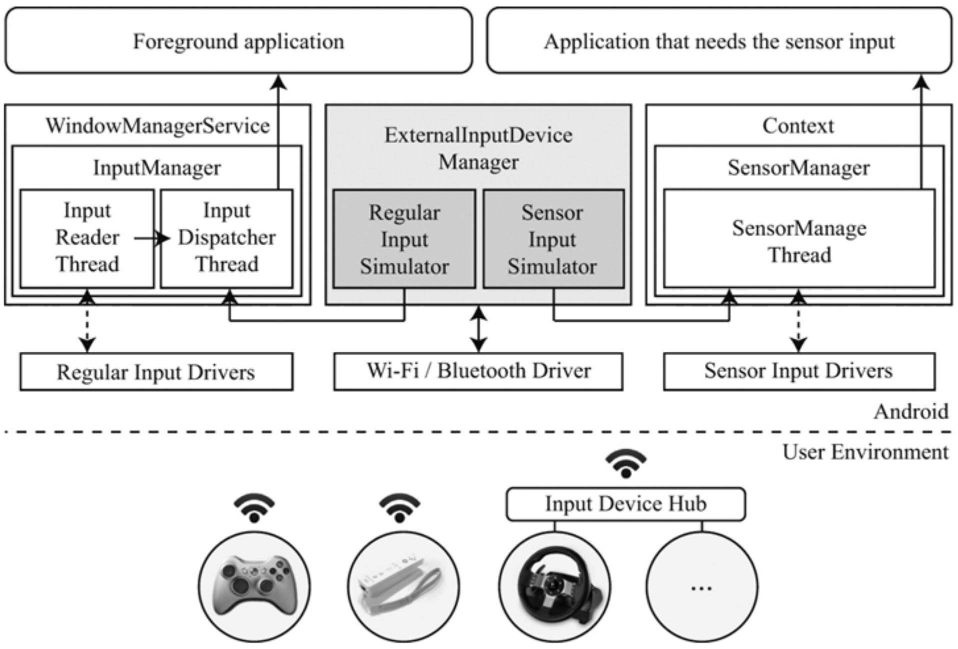

Fig. 6. Modified input delivery mechanism for adaptive input switching

For input data streams given by external input devices, ExternalInputDeviceManager maps them to any of existing input events either manually or automatically. If an external input data stream is mapped to one of regular input devices, it ishandled by "Regula Input-Simulator". That is, Regula Inpu Simulator transforms the external inputs into the mapped internal events and injects them into the queue of InputDispatcherThread. This way, the external input data can be dispatched to the current foreground application just like inputs from an existing internal input device. If mapped to one of sensor input devices, "SensorInputSimulator" handles the external input data and feeds them into SensorManage-Thread. Now, the external input data can be treated as inputs from an existing internal sensor device.

This adaptive input switching technique allows a smartphone to adaptively switch to various external input devices in a unified way. More importantly, it allows us to use external input devices for controlling general applications even if they are not designed to use those external input devices.

4.2. Use Cases of the Adaptive Input Switching

One typical use case of the adaptive input switching is to use a fancy external input device for playing a game. For example, using our adaptive input switching, a user can play the racing game with racing wheel and foot pedals as shown in **Fig. 7**. The mobile racing game in the left side of **Fig. 7** is originally designed to use the smartphone's internal accelerometer sensor's y-axis value for steering and buttons on the touch screen for acceleration or braking. Nevertheless, using the adaptive input switching technique, input data from the external steering wheel can be mapped to the accelerometer sensor's y-axis value and input data from foot pedals can be mapped to (x, y) values of buttons on the touch screen. Therefore, a user can enjoy more realistic racing game with the external input device.

Another use case we can think of is to use external input devices in a personalized way. For example, using our adaptive input switching, the disabled and elderly people can use external input devices to control the smartphone in their customized ways with the most convenient settings for their physical and mental status. This use case will be implemented in our future work.

5. IMPLEMENTATION

The proposed adaptive I/O switching technique are actually implemented on the Nexus (2010) and the Odroid-7 (2012) with Android 2.3.4 platform (Gingerbread), Linux kernel V2.6.35, Samsung Cortex A8 1GHz CPU and 512 MB RAM. The followings are our implementation results.

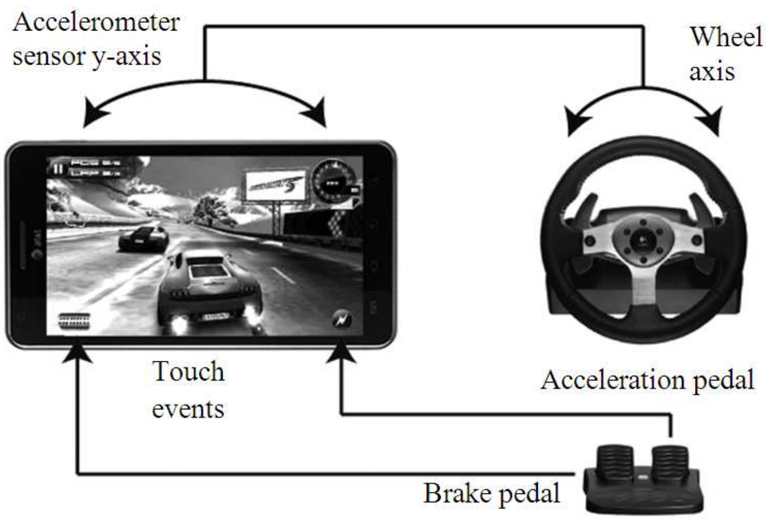

Fig. 7. Transformation of input data from the external racing wheel input device

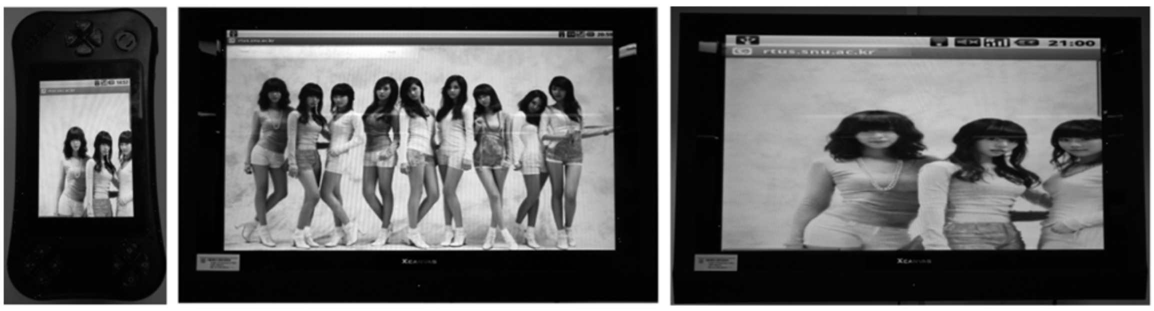

Fig. 8. Adaptive display switching (a) mobile display (b) HDTV with our resolution adaptation (c) HDTV with mirroring

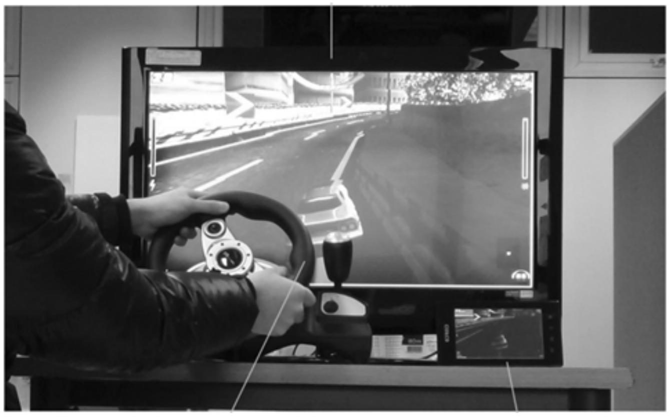

Fig. 9. Mobile racing game demonstration with the external racing wheel and HDTV

First, in order to show the effect of our adaptive display switching, **Fig. 8** shows how the mobile display content in **Fig. 8a** is displayed in HDTV with or without our resolution adaptive display switching

(**Fig. 8b and 8c**, respectively). With our resolution adaptive display switching in **Fig. 8b**, we can clearly see that the display information and framebuffer have been dynamically reconfigured adapting to the fully featured resolution of HDTV. In contrast, if we simply connect the smartphone to HDTV with the existing ad hoc solutions such as TV-out and HDMI cables, we can only see the magnified view of the same content of small LCD panel as shown in **Fig. 8c**. Moreover, this resolution adaptive display switching works for all activities including image viewer, web browser, which is not possible by an application-layer non-system solution.

Second, in order to show the combined effect of our display switching and input switching together, **Fig. 9** shows a snapshot of our system where a mobile racing game application is being played with the external racing wheel input device and the big HDTV Note that the mobile racing game application runs in the smartphone while the input comes from the external steering wheel and foot pedals and the output game image shows up in the external HDTV. More importantly, the racing game application is designed to be controlled only by smartphone’s accelerometer and touch inputs but it can be successfully controlled by external steering wheel and foot pedals thanks to our adaptive input switching technique.

6. EXPERIMENT RESULTS

In order to validate the usefulness of the proposed adaptive I/O switching, we conduct experiments with users. For these experiments, we recruit 50 evaluators. The 50 evaluators are divided into two groups, that is, each group has 25 evaluators.

Then, we ask the first group to find certain information from a web page using the small display (4 inch) of smartphone. We also ask the second group to do the same using our adaptive I/O switching to 32inch HDTV. **Figure 10** compares the time taken for the first group members and the second group members to finish their jobs. Each dot in the figure is the time for a specific evaluator to finish the job. The figure also shows the average time and 95% confidence interval of each group. Although there are large personal variations, we can clearly observe that the average time for the second group to finish implies that with our adaptive I/O switching, users can at least double their productivity on average.

Similarly, **Fig. 11** reports the times taken for another mission that is, finding differences between two similarly looking pictures. For comparison, we present two groups with same picture which has same resolution. In this different mission as well, we can observe the similar improvement of productivity.

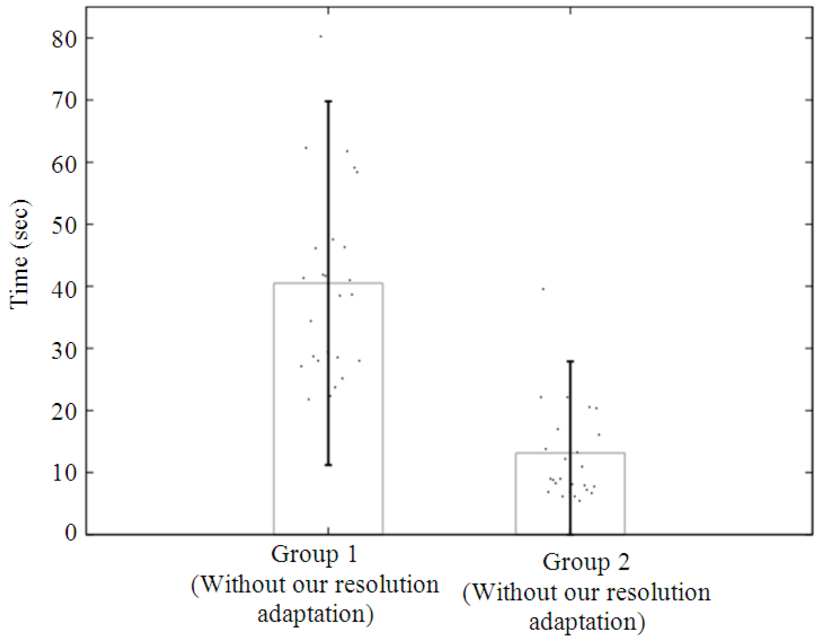

Fig. 10. Times taken for information retrieval from a web page

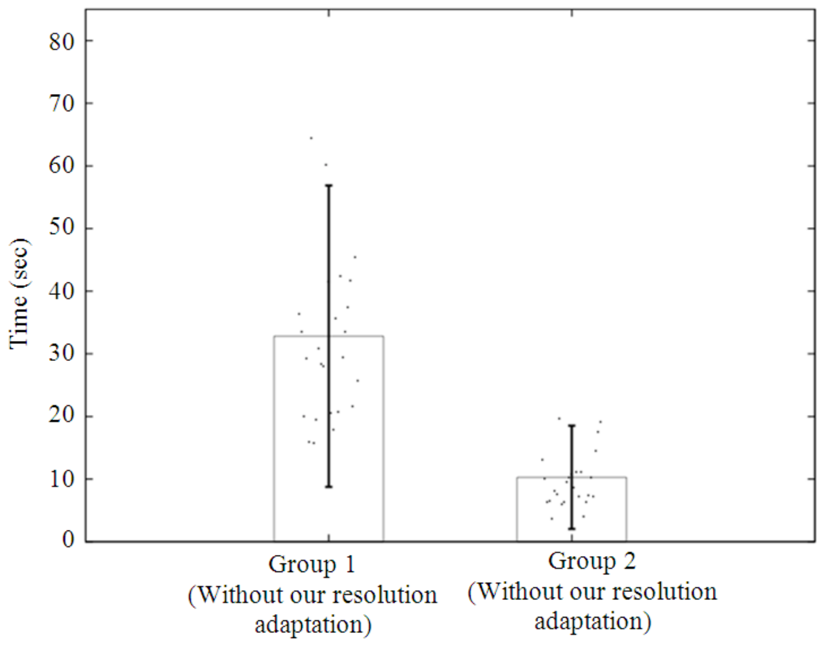

Fig. 11. Times taken for finding differences from two similar pictures

7. CONCLUSION

This study proposes our adaptive I/O switching technique as a solution for the adaptive convergence of a smartphone and external input/output devices. Unlike the existing ad hoc ways of using only specific external devices, the proposed solution can be a generic solution that can adapt to the features of various external input/output devices for general mobile applications in a unified way. The proposed solution is actually implemented and their effectiveness is demonstrated with several example use cases.

The current implementation of the modified mobile platform is limited in the sense that it supports only hard-coded mappings from external inputs to internal input events. In the future, we plan to extend our adaptive input switching such that the mobile platform can guide individual users to customize their preferred way of using external inputs.

8. ACKNOWLEDGEMENT

This research was supported by the Ministry of Science, ICT and future Planning (MSIP), Korea, under the Information Technology Research Center (ITRC) support program (NIPA-2013-H0301-13-3005) supervised by the National IT Industry Promotion Agency (NIPA).

9. REFRENCES

Android, 2013. Introducing Android: The world’s most powerful mobile platform. Google Inc., Mountain View, CA, USA.

Atrix, 2011. Motorola Atrix 4G User's Guide. Motorola Inc., pp:44-46, Schaumburg, IL, USA.

Bluetooth, 2013. Bluetooth 4.1 Features and Technical Description. Bluetooth Special Interest Group.

Citrix, 2009. Citrix Receiver for Windows User’s Guide. Citrix Systems.

Greaves, A., A. Hang and E. Rukzio, 2008. Picture browsing and map interaction using a projector phone. Proceedings of the 10th International Conference Human Computer Interaction Mobile Devices Services, Dec. 2-5, ACM, USA, pp: 527-530. DOI: 10.1145/1409240.1409336

Ha, K., K. Kang and J. Lee, 2010. N-screen service using I/O virtualization technology. Proceedings of the International Conference Information Communication Technology Convergence, Nov. 17-19, South Korea, pp: 525-526. DOI: 10.1109/ICTC.2010.5674784

HDMI, 2009. Introducing HDMI 1.4 Specification Features. HDMI LLC.

iControlPad, 2011. Gaming controls for your smartphone-instruction manual. iControlPad Inc., Newcastle, UK.

IKVM, 2010. IKVM.NET User's Guide. IKVM.

Lawton, G., 2008. WirelessHD Video Heats Up. Comput., 41:18-20. DOI: 10.1109/MC.2008.509

MHL, 2008. Whit is MHL (Mobile High-Definition Link), MHL LLC. Sunnyvale, CA, USA.

Miracast, 2013. Wi-Fi certified miracast: Extending the Wi-Fi experience to seamless video display. Wi-Fi Alliance. Austin, TX, USA.

Mobizen, 2012. Installing and launching mobizen. RSupport, Seoul, South Korea.

Mora, P. and Z. Papp, 2012. WebKey for Android. WebKey, Hungary.

MyMobiler, 2012. MyMobiler for Android-Remote Display and Input. MTUX Corp., Redmond, USA.

Nexus, S., 2010. Nexus S owner’s guide. Google Inc., Mountain View, CA, USA.

Odroid-7, 2012. Odroide-7 Platform Developer Edition. HardKernel Co., Anyang, South Korea.

OpenGL, 2013. OpenGL ES Version 3.0.2. Khronos Group, Beaverton, USA.

ORiordan, B., K. Curran and D. Woods, 2005. Investigating text input methods for mobile phones. J. Comput. Sci., 1: 189-199. DOI: 10.3844/jcssp.2005.189.199

RDP, 2013. Remote desktop protocol. Microsoft.

Richardson, T., F.Q. Stafford, R.K. Wood and A. Hopper, 1998. Virtual network computing. IEEE Int. Comput. 2: 33-38. DOI: 10.1109/4236.656066

Singh, D., R. Omar and A. Anuar, 2010. Low cost interactive electronic whiteboard using Nintendo Wii remote. Am. J. Applied Sci., 7: 1458-1463.

TeamViewer, T., 2013. TeamViewer 9 Manual-Remote Control. TeamViewer, Uhingen, Germany.

UPnP, 2010. White Paper: Universal Plug and Play (UPnP)-Your Simple Solution for Home, Office and Small Business interoperability.

WiDi, 2010. Intel Wireless Display (WiDi). Inte.

WirelessHD, 2010. WirelessHD Specification Version 1.1 Overview. WirelessHD Consortium.

Zeemote, 2012. Zeemote JS1 Mobile Gaming Controller. Zeemote Technology Inc. Santa Clara, USA.