Content-centric Display Energy Management for Mobile Devices

Dongwon Kim, Nohyun Jung, Hojung Cha
Department of Computer Science
Yonsei University
Seoul, Korea
{dwkim,nhjung,hjcha}@cs.yonsei.ac.kr

ABSTRACT
While active studies have been conducted to reduce the power consumption of display-related components of mobile devices, previous work has rarely approached the issues without having to deteriorate graphical quality. In this paper, we propose an effective scheme to reduce display energy consumption without compromising user experience. We first define a metric called the content rate from which an appropriate refresh rate is determined for displaying content. The proposed system then sets an optimal refresh rate based on the content rate. Extensive experiments demonstrate that our system effectively reduces the total power in commercial smartphones, yet the display quality is satisfactorily maintained.

Categories and Subject Descriptors
I.4.3 [Image Processing and Computer Vision]: Enhancement; B.4.2 [**Input/Output Devices**]: Image display

General Terms
Management, Performance, Experimentation

Keywords
Power Management, Refresh Rate, Smartphone

1. INTRODUCTION
The energy consumption of smartphones is rapidly increasing, as applications demand more performance [1]. In particular, the display-related subsystem significantly contributes to the energy consumption of mobile devices [2]. Active work has recently been conducted to lessen the power consumption of displays. Prior work has reduced the display power using dynamic voltage scaling (DVS) of the display [3-4]. Other studies have attempted power reduction by making use of LCD or OLED display characteristics [5-7]. However, previous work has tended to compromise the display quality when power consumption has been reduced; hence, display power issues have rarely been handled effectively without having to deteriorate graphic quality.

Meanwhile, most mobile applications scarcely shoot in 60fps of the frame rate[1], whereas commercial smartphones normally use a fixed rate of 60Hz for the refresh[2]. Applications with a high frame rate tend to update frames frequently, even when there are no changes in display content. Consequently, the display subsystem continuously wastes power to update the unchanged frames. A new scheme is therefore required to eliminate the power consumption caused by redundant graphic operations in mobile devices.

In this paper, we propose a display power management scheme that efficiently reduces redundant power consumption without having to compromise user experience. We define a metric called the *content rate,* which means the frequency of essential frame updates that does not result in redundancy in frame contents. The proposed system first detects the content rate for each application efficiently, and then reduces the unnecessary power consumption by controlling the refresh rate without deteriorating the graphic quality. We implemented the system on real mobile devices and experimentally validated its effectiveness with commercial applications.

2. BACKGROUND AND MOTIVATION
We explain the graphic display operation on mobile devices and then investigate the redundancy issue on frame updates in commercial applications.

2.1 Graphic Displaying
We first describe several terms regarding graphic display operation: the frame rate, the refresh rate, and the vertical synchronization technique (V-Sync). To display an image on the screen, Android uses *Surface Manager,* which combines the rendered partial images (i.e., surfaces) of applications and then updates the frames by writing a combined image on the framebuffer to draw it on screen. The frame rate indicates the frequency of the frame updates conducted by *Surface Manager*. The refresh rate is the frequency of display hardware updates made using framebuffer. Thus, the frames outnumbering the refresh rate are redundant, as those frames over the refresh rate are not drawn on screen. To handle this problem, V-Sync [8] limits the frame rate lower than the refresh rate. Figure 1 illustrates that the framebuffer is updated with *Surface Manager* in Android, while the screen is refreshed with the display hardware.

The current Android platform fixes the refresh rate as 60 Hz and employs V-Sync. The fixed refresh rate can, however, lead to energy waste when the frame rate is lower than the refresh rate. In the following section, we investigate the frame rates of commercial Android applications to motivate our work.

*DAC 'I*4, June 01 - 05 2014, San Francisco, CA, USA
ACM 978-1-4503-2730-5/14/06$15.00.

[1] The frame rate (frames per second or FPS) is the rate at which an imaging device generates consecutive images.

[2] The refresh rate is the rate at which the display hardware refreshes the images called frames.

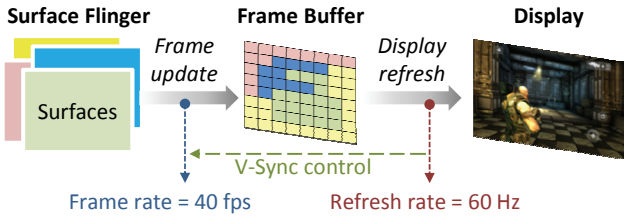
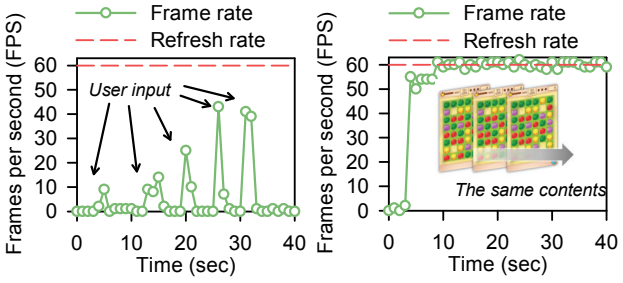


Figure 1. Graphic display procedure in Android

2.2 Frame Rate and Refresh Rate

The frame rate depends on the frequency of frame updates requested by an application. Figure 2 shows, for example, the traces of frame rate with Facebook [9] and Jelly Splash [10]. In Figure 2(a), the frame rate of Facebook is low most of the time, except when user requests occur. On the other hand, Jelly Splash in Figure 2(b) remains at about 60 fps most of the time, even when the content of frame is not changed. Based on these results, we claim two observations. First, the display hardware redundantly refreshes the screen even when the frame rate is low, as the refresh rate of display is fixed at 60 Hz. Second, some applications frequently request frame updates although the display content does not change. Thus, we may reduce the power consumption of applications by controlling the refresh rate dynamically to a minimum. To generalize our observations, we have further investigated the frame rates of common Android applications.

For the experiment, we chose 30 commercial applications in Google Play Top Charts South Korea—15 game applications and 15 general applications—and ran them for approximately 3 minutes on Samsung Galaxy S3s (SHV-E210S). In addition to measuring the frame rate, we also measured the *meaningful frame rate* which excludes redundant frames. (Section 3.1 details the scheme to acquire the meaningful frame rate of an application.) Figure 3 shows both the meaningful frame rate and the redundant frame rate for general applications and game applications. Most of the general applications require less than 30 fps, since most applications constantly maintain an image on screen, as illustrated in Figure 3(a) and (c). However, about 40% of them exhibit approximately 20 fps of the redundant frame rate (e.g., Cash Slides, Daum Maps), as shown in Figure 3(d). While Figure 3(b) and (c) shows that all the game applications update the display at more than 30 fps, 80% of them have more than 20 redundant frames per second, as presented in Figure 3(d). Hence, some of



(a) Facebook **(b) Jelly Splash**

Figure 2. Frame rate and refresh rate traces

the general applications and most of the games redundantly request frames updates.

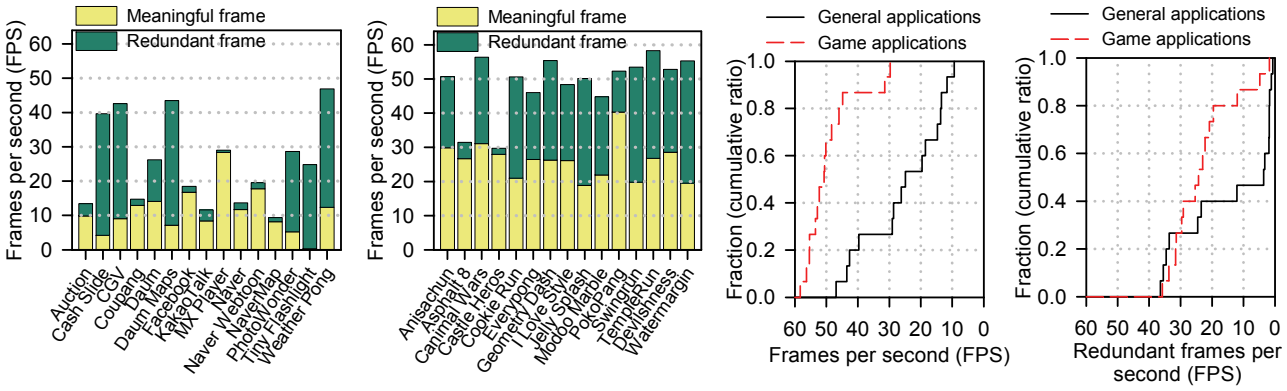
Our preliminary study indicates that the fixed refresh rate superfluously consumes the energy of smartphones. Additionally, some applications waste graphical resources through redundant frame updates. To solve these problems, we have designed a system that reduces the display-related power consumption by managing the refresh rate effectively with a given frame rate of an application.

3. CONTENT-CENTRIC DISPLAY POWER MANAGEMENT

To achieve the goal of reducing the display power consumption, the proposed system accurately measures the content rate at a low cost using two techniques: double buffering and grid-based comparison techniques. In addition, we have devised a section-based control and touch boosting techniques to manage the refresh rate efficiently, according to the content rate.

3.1 Measuring Content Rate

To overcome the problem presented in Section 2, a new scheme is required to measure the rate of meaningful frames. We now define a metric called the *content rate,* which is the number of contents per second. The content rate is calculated by subtracting the redundant frame rate from the frame rate, and it is measured by monitoring the framebuffer as follows. When *Surface Manager* updates the framebuffer, the framebuffer data are stored at an extra buffer. Thereafter, when *Surface Manager* updates the framebuffer again, the content of current framebuffer is compared with that of the prior frame contained in the extra buffer. In this



(a) General applications **(b) Game applications** **(c) Application (FPS)** **(d) Application (Redundant FPS)**

Figure 3. Redundancy in frame rate for commercial applications

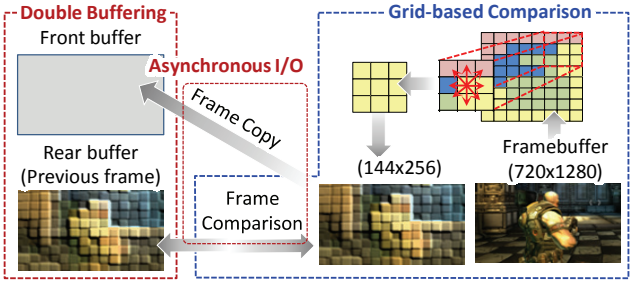


Figure 4. Example of double buffering and grid-based comparison techniques with Galaxy S3 (SHV-E210S)

way, we count the meaningful frames that only contain unique content.

Double Buffering To compare prior and current framebuffers, an extra buffer is needed to store previous framebuffer. In this phase, we reduce the comparison cost by using a double buffering technique [11]. This technique minimizes the operational delay by asynchronous I/O with front and back buffers. Figure 4 illustrates the scheme. While the I/O operations are not simultaneously processed with a single buffer, the double buffering technique improves the performance of measuring the content rate by allowing a continuous operation.

Grid-based Comparison The cost of content rate estimation depends on the display resolution, that is, the number of pixels that are stored in a framebuffer. Due to the high display resolution of modern smartphones, direct comparison of buffers is highly burdensome. In order to minimize the cost, our system compares the partial region of two buffers by using a grid-based comparison scheme, where the RGB data of the grid are regarded as the center pixel of each grid, as illustrated in Figure 4. This technique enables the proposed system to measure the content rate at almost no cost.

3.2 Refresh Rate Control

Section-based Control The refresh rate levels depend on the display hardware characteristics of the target device. To develop the refresh rate control, we used Samsung Galaxy S3, which has five available refresh rates: 60 Hz, 40 Hz, 30 Hz, 24 Hz, and 20 Hz. In our initial attempt, we first tried to adjust the refresh rate to the current content rate. For example, if the content rate exceeds 20 fps, the system increases the refresh rate to 24 Hz. However, this algorithm did not work adequately, since the content rate cannot exceed the refresh rate due to the V-Sync mechanism, as explained in Section 2.1. In other words, the system cannot measure a content rate higher than 20 fps after the refresh rate is adjusted to 20 Hz. Therefore, the control algorithm should keep the refresh rate higher than the content rate. Accordingly, we developed a section-based control technique that relies on a predefined section table to match the content rate with the corresponding refresh rate. In our scheme, the section table is constructed by Equation (1).

$$\beta_i = \begin{cases} \frac{\gamma_{i-1}+\gamma_i}{2} & if\ i > 1 \\ \frac{\gamma_i}{2} & if\ i = 1 \end{cases}, 0 < i < n \qquad (1)$$

where i is the index of the refresh rates, β_i is the threshold which splits the i-th and $(i+1)$-th sections, and γ_i is the i-th refresh rate. Thus, the range is split by the median between adjacent refresh

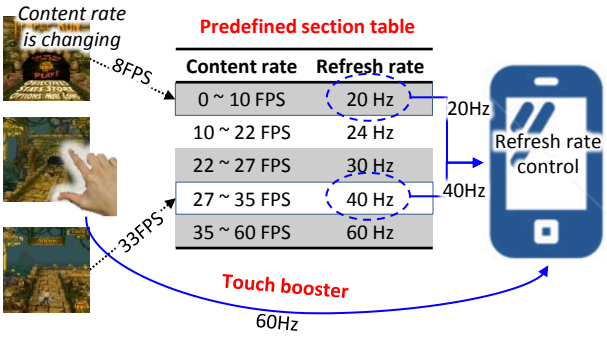


Figure 5. Example of section-based control and touch boosting techniques with Galaxy S3 (SHV-E210S)

rates. Note that the thresholds should be redefined when the available refresh rates are changed.

Figure 5 shows an example illustrating the technique. The application initially updates frames at 8 fps of the content rate upon which the refresh rate is set to 20 Hz according to the predefined section table. When the application displays the screen at 33 fps of the content rate, the refresh rate is adjusted to 40 Hz accordingly.

Touch Boosting As shown in Figure 2(a), the frame rate of application rapidly increases with user interaction However, the section-based control technique would not be responsive to a sudden change in the frame rate, because the frame rate cannot exceed the refresh rate. To eliminate the delay of section-based control, we developed a touch boosting technique that enforces the refresh rate, regardless of the content rate, to maximum when a touch event occurs. The concept is illustrated in Figure 5.

4. EVALUATION

We evaluate the proposed schemes in four parts. The first and second parts validate the main functionalities of the proposed system, metering the content rate and controlling the refresh rate. Then, we evaluate the proposed system's overall performance in terms of graphic quality and power reduction.

For the experiment, we used Galaxy S3 LTE (SHV-E210S) with a kernel modification to enable refresh rate control at runtime. The hardware supports five refresh rate levels as explained earlier. We evaluated the proposed system with the commercial applications mentioned in Section 2.2. For each application, we repeated the same script generated by Monkey [12], and measured the device's power consumption with and without the proposed system. The device's power was measured using a Monsoon power meter [13] with the device's screen brightness at 50%.

4.1 Accuracy of Content Rate Monitoring

We evaluated the accuracy and the computational overhead of the content rate metering scheme with 30 commercial applications. Figure 6 shows the time taken for the scheme according to the number of compared pixels. When all pixels are examined, the maximum run time is more than 40 ms. In the case of 36K and 9K pixels, the run time is about 9 ms and 5 ms, respectively. The content rate metering with less than 9K pixels takes less than 1 ms. Note that examining all pixels on the screen is not practical, since the job cannot be completed for the maximum frame rate of 60 Hz (i.e., within 1/60 seconds=16.67 ms). Hence, the content rate should be measured with fewer than 36K pixels in our experimental setup.

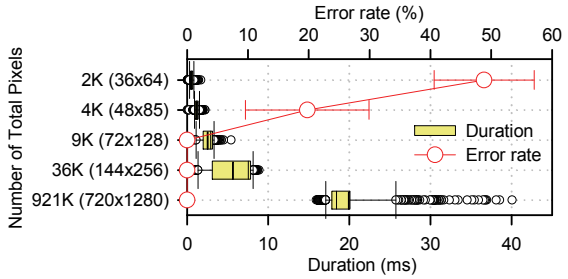


Figure 6. Accuracy of the content rate vs. pixels

For the accuracy analysis, we ran several live wallpaper applications that continuously display the consecutive images on the screen with the frame rate below 25 fps. The accuracy of our scheme was initially 100 %, since the image on the screen significantly changes at each frame. Hence, we configured an extreme environment using Nexus Revampled live wallpaper [14] that continuously makes small changes by moving small dots across the screen. Figure 6 shows the accuracy of the content metering according to the number of pixels. The content rate estimation was accurate with more than 9K pixels. Based on the results, we conclude that content rate metering with 36K or 9K pixels is adequate with almost no computational overhead in our experimental setup.

4.2 Validation of Refresh Rate Control

We now validate the section-based control and touch boosting techniques that are used for controlling the refresh rate. For the experiment, we used Facebook and Jelly Splash again and ran them repeatedly to estimate the refresh rate as well as the content rate. Figure 7(a) and (c) show the traces with the section-based control technique only, whereas Figure 7(b) and (d) show the trace of the content rate and the refresh rate with both techniques applied.

As shown in Figure 7(a) and (c), the content rate slowly increases with changes on the screen. Accordingly, the refresh rate is adequately adjusted by section-based control. However, there are several cases where the refresh rate is lower than the actual content rate, particularly in the touch event. This may cause degradation in graphic quality as a result of frames being dropped.

Figure 7(b) and (d) show the result of the refresh rate control with the touch boosting scheme applied. A large fluctuation in the refresh rate is observed due to the touch boosting scheme. Compared to Figure 7(a) and (c), the occurrence of frame dropping is significantly reduced. This means that the touch boosting technique indeed maintains the graphic quality by quickly responding to the rapid increase in the content rate.

4.3 Power Save

In order to validate the power-saving effect of the proposed mechanism, we first evaluated the system with the same applications, Facebook and Jelly Splash, used in the preliminary experiment. We compared the device's power consumption with and without the proposed system, repeating the same script generated by Monkey [12].

Figure 8 shows the results, which are calculated by subtracting the power consumption of the proposed system from the original one. For both cases, our system has, indeed, reduced power consumption. The average power saved using section-based control is about 150 mW (±120 mW) and 500 mW (±150 mW), respectively. The amount of power saved with Jelly Splash is

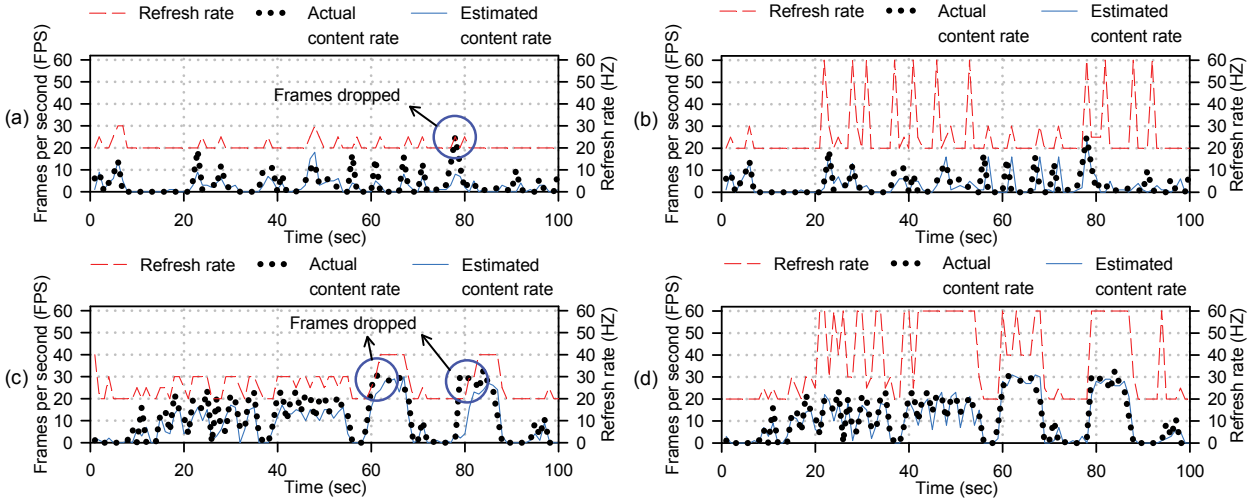


Figure 7. Trace of the content rate and refresh rate with section-based control and touch boosting schemes, (a) Facebook with section-based control, (b) Facebook with section-based control and touch boosting, (c) Jelly Splash with section-based control, (d) Jelly Splash with section-based control and touch boosting.

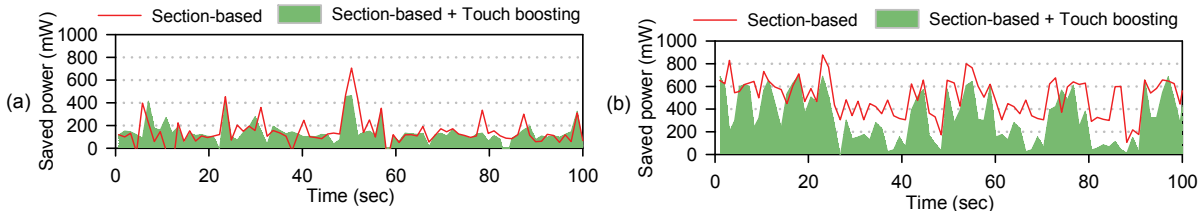


Figure 8. Power save with section-based control and touch boosting methods, (a) Facebook, (b) Jelly Splash

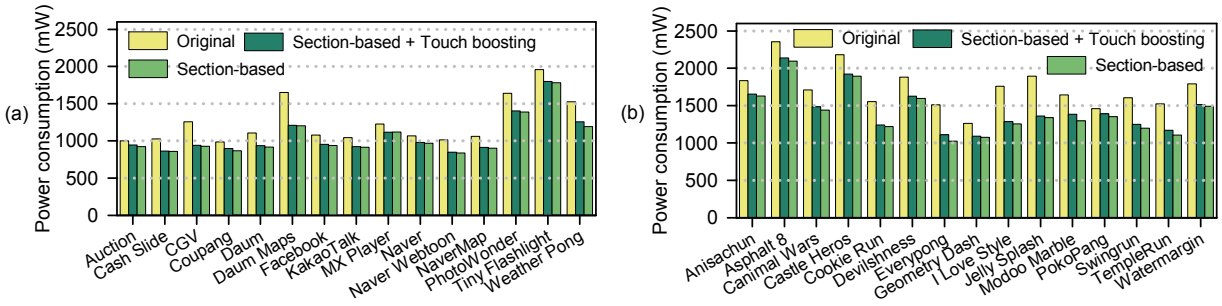


Figure 9. Power-saving effect, (a) General applications, (b) Game applications

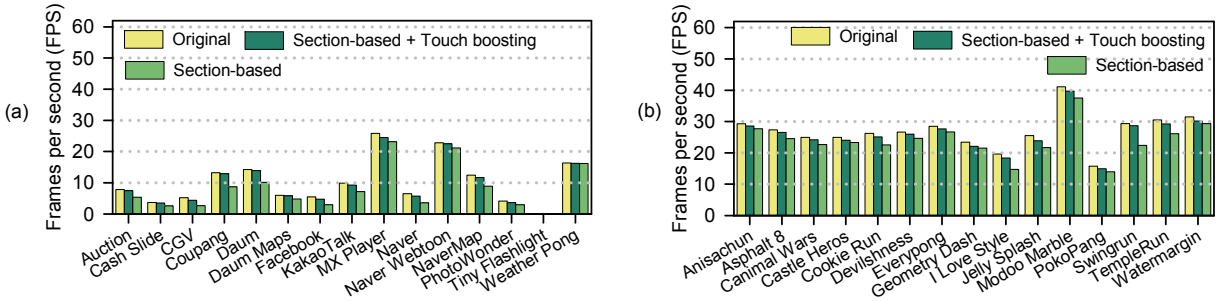


Figure 10. Effect on content rate, (a) General applications, (b) Game applications

much larger than that of Facebook, since Jelly Splash keeps a high frame rate of almost 60 fps regardless of the content rate, as shown in Figure 2(b). The average power saved including the touch boosting scheme is 135 mW (±90 mW) and 330 mW (±220 mW). The amount of saved power is slightly reduced by the touch boosting scheme, but this process is required to maintain the graphic quality.

We further conducted the same experiments with the 30 commercial applications mentioned in Section 2.2. Figure 9(a) and (b) shows that the average power reductions of general applications and game applications are about 120 mW and 290 mW, respectively. Note that the proposed system significantly reduced the power consumption of the game applications as well as several general applications (i.e., CGV, Daum Maps) that generate a large number of redundant frames.

The proposed system reduces the power consumption of general and game applications to a maximum of 440 mW and 530 mW, respectively. For 80% of general and game applications, the power reduction is more than 110 mW and 220 mW. The power consumption with the touch boosting scheme is slightly increased by about 16 mW and 30 mW for general and game applications, but this is not significant.

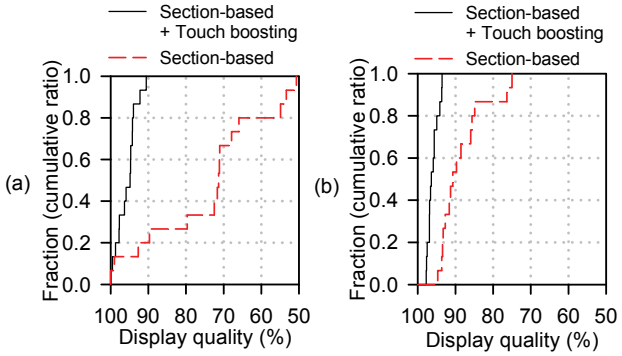


Figure 11. Display quality: (a) General applications, (b) Game applications

In summary, the proposed system significantly reduces the amount of power consumed by eliminating the redundant frames.

4.4 Graphic Quality Analysis

Due to refresh rate control, the degradation of graphic quality can be caused by unexpected frame dropping when the estimated content rate is lower than the actual content rate. In order to assess this issue, we compared the content rate of the proposed system with the actual content rate. Figure 10 shows the frame rate of 30 applications for various experimetal setups. The refresh rate control with the touch boosting scheme estimates the content rate to be approximately the same as the actual content rate. However, the content rate is underestimated without the touch-boosting technique, as a user's touch event involves changes on the screen. The number of frames dropped with section-based control is less than 2.9 frames and 3.8 frames per second for 80 % of both general and game applications. The results are not, in fact, satisfactory, since users may feel uncomfortable with more than 3 fps of frame dropping. In contrast, the number of frames dropped with the touch-boosting scheme is less than 0.7 fps and 1.3 fps for 80% of general and game applications, which results in virtually no degradation in graphic quality.

Figure 11 shows the display quality which is calculated by dividing the estimated content rate by the actual content rate, as presented in Figure 10. The display quality with section-based control is maintained in more than 55 % and 85 % for 80 % of general and game applications, respectively. Since the section-based control was not responsive to a sudden change in the content rate, a large amount of dropped frames deteriorated the display quality as presented in Figure 10. In constrast, the display quality with the touch boosting technique is maintained in more than 95 % for 80 % of both general and game applications. Since the touch boosting technique prevents frames from being dropped, the display quality is actually well preserved. Consequently, the proposed system ensures that the display quality is maintained in more than 90 % for all of the applications.

Table 1. Power-saving effect and display quality

Application type	Method	Saved power (%)	Display quality (%)
General applications	Section-based	18.6 (±8.93)	74.1 (±15.6)
	Section-based +Touch boosting	16.73 (±8.74)	95.7 (±2.7)
Game applications	Section-based	25.13 (±12.36)	88.5 (±6.0)
	Section-based +Touch boosting	21.25 (±10.70)	96.0 (±1.4)

Table 1 summarizes the proposed system's performance change in terms of the saved power and the display's quality change. The section-based refresh rate control significantly reduced the display power consumption by eliminating redundant frames. The touch-boosting scheme induces slight power increment, but compensates for the graphic quality degradation by quickly responding to the sudden rise in the frame rate caused by a touch event.

5. RELATED WORK

The display subsystem's power consumption has been a key issue in power management research. Many approaches using dynamic voltage scaling (DVS) have been proposed to reduce power consumption while minimizing luminance degradation. [3-4,15]. Active studies have also been conducted to utilize the power characteristics of OLED displays where power consumption is greatly affected by the screen color. Chameleon [7] reduced the power consumption of displays by calibrating the color of the web site automatically. Focus [5] inferred the region of interest (ROI) where users are assumed to focus on, which makes the rest of the region dark. Anand [6] reduced LCD power consumption by adjusting the backlight and gamma level appropriately. According to Han et al. [16], the scrolling operation increases power consumption significantly. They reduced power consumption using adaptive frame rate control that is responsive to the scrolling operation.

Prior work has mostly resulted in a reduction in graphic quality, since the schemes have focused more on power reduction rather than retaining graphic quality. In comparison, our work retains graphical quality while reducing superfluous power consumption by eliminating redundant graphical operations effectively.

6. CONCLUSION

This paper proposed a refresh rate management system that efficiently reduces power consumption without deteriorating the graphic quality in mobile devices. We defined a content rate, which is the number of meaningful frames with redundant frames excluded. Our system accurately measures the content rate with a low cost at runtime with double buffering and grid-based comparison methods. Furthermore, the section-based control and touch boosting techniques adequately adjust the refresh rate according to the content rate. The extensive experiment with 30 commercial applications validates that the system makes about 230 mW of power reduction and 95 % of quality maintenance on average.

7. ACKNOWLEDGMENTS

This work was supported by a grant from the National Research Foundation of Korea (NRF), funded by the Korean government, Ministry of Education, Science and Technology under Grant (No.2011-0015332).

8. REFERENCES

[1] Carroll, Aaron, and Gernot Heiser. An analysis of power consumption in a smartphone. *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, 2010.

[2] Carroll, Aaron, and Gernot Heiser. The systems hacker's guide to the galaxy energy usage in a modern smartphone. *Proceedings of the 4th Asia-Pacific Workshop on Systems*. ACM, 2013.

[3] Chen, Xiang, et al. Quality-retaining OLED dynamic voltage scaling for video streaming applications on mobile devices. *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*. IEEE, 2012.

[4] Chen, Xiang, et al. Fine-grained dynamic voltage scaling on OLED display. *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*. IEEE, 2012.

[5] Tan, Kiat Wee, et al. FOCUS: a usable & effective approach to OLED display power management. *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2013.

[6] Anand, Bhojan, et al. Adaptive display power management for mobile games. *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services.* ACM, 2011. Mobisys Chameleon

[7] Dong, Mian, and Lin Zhong. Chameleon: a color-adaptive web browser for mobile OLED displays. Mobile Computing, *IEEE Transactions* on 11.5 (2012): 724-738.

[8] Google, Butter project, https://developers.google.com/events/io/2012/sessions/gooio2012/109/

[9] Facebook, Facebook, https://play.google.com/store/apps/details?id=com.facebook.katana

[10] Wooga, Jelly splash, https://play.google.com/store/apps/details?id=com.wooga.jelly_splash

[11] Oracle, Double buffering and page flipping, http://docs.oracle.com/javase/tutorial/extra/fullscreen/doublebuf.html

[12] Android, Monkey, http://developer.android.com/tools/help/monkey.html

[13] M.S. Inc., Monsoon Power Monitor, http://www.msoon.com/LabEquipment/PowerMonitor

[14] Stealthcopter, Nexus Revampled, https://play.google.com/store/apps/details?id=com.stealthcopter.nexusrevamped

[15] Shin, Donghwa, et al. Dynamic voltage scaling of OLED displays. *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*. IEEE, 2011.

[16] Han, Haofu, et al. E 3: energy-efficient engine for frame rate adaptation on smartphones. *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2013