SOFTWARE—PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 2006; (in press)
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/spe.753

Dynamic refresh-rate scaling via frame buffer monitoring for power-aware LCD management

Hyoseung Kim[1], Hojung Cha[1,*,†] and Rhan Ha[2]

[1]*Department of Computer Science, Yonsei University, Seodaemun-gu, Shinchon-dong 134, Seoul 120-749, Korea*
[2]*Department of Computer Engineering, Hongik University, Mapo-gu, Sangsoo-Dong 72-1, Seoul 121-791, Korea*

SUMMARY

In recent years, there has been wide-spread use of large and high-resolution liquid-crystal displays (LCDs) on handheld devices. The portion of LCD power consumption in the overall system has gradually increased. While most of the previous research on LCD power management has focused on the hardware level, practical mechanisms at the software level are hardly known. This paper presents a power-aware LCD management mechanism, based on dynamic refresh-rate scaling and frame buffer monitoring. The proposed mechanism guarantees the display quality of service, which is inherently specified by content types. The mechanism does not require additional hardware or modifications to applications. The experiment results—on a commercial PDA with a 320 x 240 resolution—show that the proposed mechanisms effectively reduce the power consumption by up to 10%, while satisfying the display quality requirements for the LCD screen. Copyright © 2006 John Wiley & Sons, Ltd.

Received 22 November 2004; Revised 5 April 2006; Accepted 5 April 2006

KEY WORDS: LCD power management; frame buffer monitoring; dynamic refresh-rate scaling

1. INTRODUCTION

Computing environments have been continuously moving toward mobile and embedded platforms. Mobile embedded systems, such as PDAs and smart phones, are battery-operated. While the computational power of the devices has increased rapidly, the lifetime of batteries is not expected

*Correspondence to: Hojung Cha, Department of Computer Science, Yonsei University, Seodaemun-gu, Shinchon-dong 134, Seoul 120-749, Korea.
†E-mail: hjcha@cs.yonsei.ac.kr

Contract/grant sponsor: Korea Science and Engineering Foundation (National Research Laboratory Program); contract/grant number: 2005-01352
Contract/grant sponsor: IITA ITRC Programs (MMRC, HY-SDR)
Contract/grant sponsor: City of Seoul, Korea, Seoul R&BD Program

to improve drastically in the near future. Active research has recently been conducted to extend battery lifetimes, especially on the software side. For instance, DPM [1] and DVS [2] are representative software-based research efforts to reduce the power usage of the system, by changing the power level of the underlying hardware dynamically. Meanwhile, as multimedia applications on large and high-resolution liquid-crystal displays (LCDs) become popular on handheld devices, the necessity of LCD power management becomes even more apparent. Udani and Smith [3], for example, have reported that the LCD display component can easily account for over 50% of total energy consumption; this implies the importance of the development of effective power management mechanism for LCD devices.

Previous works on the software approach to LCD power management have focused on device control techniques, but the display QoS (quality of service) has not been seriously considered. Choi *et al.* [4] introduced various techniques for low-power thin film transistor (TFT)-LCD management, such as variable duty-ratio refresh, dynamic color-depth control, and brightness compensation and contrast enhancement with backlight luminance dimming. These techniques, however, require additional hardware and hence are difficult to apply directly to existing mobile systems. Gatti *et al.* [5] has improved the previously known TFT-LCD power techniques by suggesting techniques such as variable dot clock, variable frame refresh, liquid crystal orientation shift and backlight auto-regulation. The techniques cover a wide range of power management possibilities for LCD, but they also have limitations due to additional hardware requirements or window-based graphical user interface (GUI) environments. Cheng *et al.* [6] introduced a bit transition technique called palette encoding, which reduces the bandwidth of the system bus. The encoding overhead is, however, not trivial and the method can express only 256 colors per frame, thereby degrading the display quality. Zhong and Jha [7] suggested a power-aware GUI system, but it requires modifications to the existing GUI systems, and users should therefore learn the new interface for effectively using the system. Flinn and Satyanarayanan [8] proposed a zoned backlighting technique that divides the overall backlight area into multiple zones and selectively turns the zones on and off. The mechanism, however, has a practical limitation due to the lack of hardware support of the commercially available LCDs. Dalton and Eills [9] used a technique to turn off the display while user is away from the device, by recognizing the user's face by camera. Iyer *et al.* [10], Harter *et al.* [11] and Bloom *et al.* [12] presented a low-power technique for organic light-emitting diode (OLED) devices; the technique reduces the quality of non-focused windows by dimming or not displaying them on the screen. The technique has a weakness similar to that of Gatti's method and, moreover, the technique cannot be applied to the currently prevailing TFT-LCD devices. Meanwhile, Kamijoh *et al.* [13] showed an interesting result that on the OLED-based wrist-watch computer [14] the energy consumption can be reduced by controlling the font size and the width of the clock hands on the screen adequately. Shim *et al.* [15] and Cheng *et al.* [16] proposed techniques to change the contrast and brightness of the LCD dynamically, in order to compensate for the quality degradation caused by the reduction of backlight power. Cheng *et al.*'s work is limited to the grayscale LCDs, and Shim *et al.*'s work requires non-trivial processing overhead to modify the images in the frame buffer if not using additional hardware. Brakmo *et al.* [17] studied an energy-aware mechanism to display static images on the screen while the processor is in the sleep state.

Although diverse research efforts have been conducted to provide power-aware LCD management at the software level, most results are not applicable to real systems because they require changes to existing systems; more importantly, they lack policies that specifically consider the display QoS.

DOI: 10.1002/spe

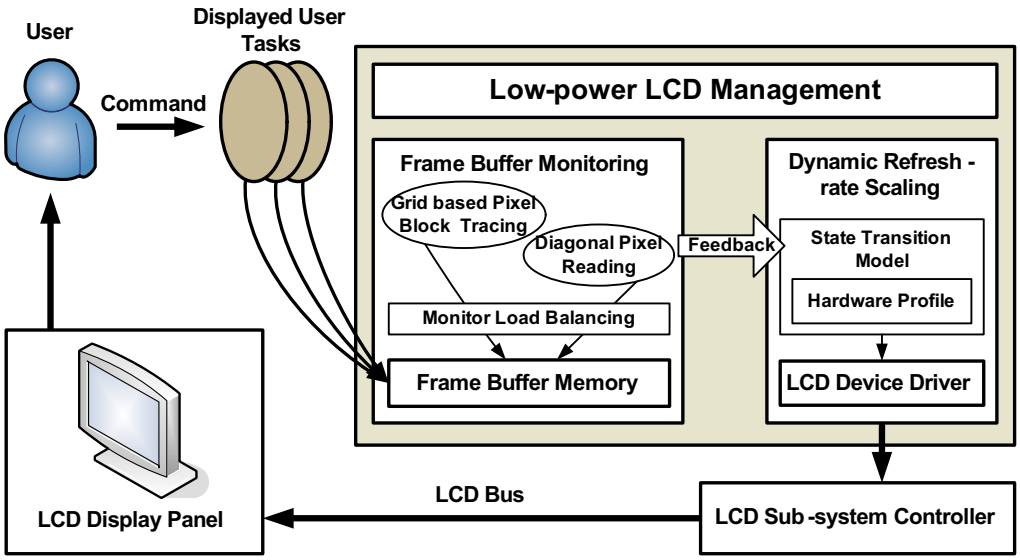


Figure 1. System overview of the proposed power-aware LCD management.

It is, therefore, essential to develop a practical yet efficient power-management mechanism for LCD devices. This paper presents a power-aware LCD management technique that guarantees the QoS of display with minimal system overhead. The mechanism does not require additional hardware or modifications to applications. The proposed mechanism analyzes the current display contents via a technique called frame buffer monitoring (hereafter referred to as FBMon) and then, depending on the display content, the power consumption of LCD device is controlled dynamically by the dynamic refresh-rate scaling (DRS) technique. The effectiveness of the proposed techniques is validated by extensive experiments conducted on a mobile device running the Linux operating system.

2. DYNAMIC REFRESH-RATE SCALING BASED ON FRAME BUFFER MONITORING

This section describes a power-aware LCD management mechanism that dynamically scales the refresh rate of LCD devices, by monitoring the display content of the graphic frame buffer in the kernel.

2.1. System overview

Figure 1 illustrates the system organization of the proposed power-aware LCD management. The system basically consists of two core modules: FBMon and DRS. FBMon monitors the frame buffer, while DRS dynamically performs power control based on feedback from FBMon.

DOI: 10.1002/spe

Table I. Effects on refresh rate control.

Application	Refresh rate (Hz)	Power consumption (Watts)	Power gain (%) over 60 Hz	Flickering
Text editor	10	1.13	10.3	No
	30	1.19	5.5	No
	60	1.26	—	No
MPEG player	10	1.49	6.3	Apparent
	30	1.52	4.4	Occasional
	60	1.59	—	No

One popular technique in controlling the LCD subsystem for power-aware applications is via backlight control. This technique, however, requires sophisticated software and hardware mechanisms to control the luminance intensity of the backlight; otherwise, the display quality degrades drastically. In our work, we adopt a rather simple mechanism for controlling the dynamic refresh rate in LCD power management. DRS adjusts the refresh rate of LCD controller to reduce power consumption of LCD systems. This mechanism can be applied directly to real systems without additional hardware equipment or modifications to existing applications; it is suitable for achieving power management dynamically. In general, a low refresh rate slows down the display signal in the LCD subsystem, and thus power reduction occurs on the display system. DRS is a finite-state machine and the state transition is conducted depending on the 'hardware profile', which in turn defines the pre-defined refresh-rate information of the underlying hardware.

FBMon is a frame buffer monitoring mechanism that is developed for use with DRS, in order to provide quality of display. In our work the display QoS is defined to be met if the end-user does not experience the flickering effect as the result of the DRS operation. To guarantee the display quality, the refresh rate is reduced only to the point where flickering ceases on the LCD panel. However, due to the characteristics of a LCD device, the minimum refresh rate differs according to the display contents. Table I shows experiment results of power reduction versus flickering level, for various refresh rate changes on a Compaq iPAQ3860 PDA running two types of application: a text editor and an MPEG video player. Concerning the classification of the flickering level, we have not conducted any formal evaluation on flickering, but rather used a subjective user study. We have defined three types of flickering—'apparent', 'occasional', 'no'—for the benchmarks used in the experiment. For 'apparent' and 'no' cases, everyone and none of them, respectively, participated in the experiment experienced flickering for the given set of benchmarks. The split decision among the participants regarding the occurrence of flickering was categorized as 'occasional'. The experiments reveal that, for text editing, no flickering is experienced until the 10 Hz refresh rate, whereas flickering is noticeable at 10 Hz when video runs on an LCD. This means that information on currently displayed contents is necessary in order to achieve power reduction without degrading the display quality. FBMon is developed to collect this information, and the following sections describe the details of FBMon and DRS.

DOI: 10.1002/spe

2.2. FBMon

Since the frame buffer within the kernel stores the contents of a graphic display, the characteristics of the display screen can be obtained by simply examining the frame buffer. FBMon extracts the approximate ratio of content display area on LCD by monitoring the frame buffer. The ratio of image is defined as the portion of high-quality content area divided by the area of the entire screen. As high-resolution images and videos require high display quality, the refresh-rate control should be applied carefully. Hence, FBMon first detects the ratio of high-quality image, and depending on its value, the refresh rate is dynamically determined.

The mechanism to analyze the ratio of high-quality image to the entire displayed contents is as follows. The displayed, on-screen contents are divided into two areas: high-quality image or video area, and the remaining area. The remaining area normally consists of low-quality images or text, so the display quality for this area is hardly affected by a low refresh rate in DRS operations. Here, we call the high-quality image or video area the *image area*, and the remaining portion the *text area*. FBMon scans the frame buffer by color distribution, and then classifies the pixels as either image or text areas. The entire screen is divided into *pixel blocks*, where each pixel block consists of $k \times k$ pixels. A pixel block is specified by the distribution of colors used in the block. For example, the number of different colors in a pixel block that belongs to the image area is much larger than that of the text area. Hence, with a threshold value set on the number of colors, a pixel block is practically classified as being of either an image area or text area.

Since the number of pixels to be examined is not small, a fast and efficient method is required for examining the frame buffer effectively. We have devised three techniques for this purpose: *grid-based pixel block tracing*, *diagonal pixel reading* and *monitor load balancing*. Both grid-based pixel block tracing and diagonal pixel reading reduce the number of pixels to be inspected, based on the pixel information of display contents. Monitor load balancing, on the other hand, is a technique for distributing a monitoring load evenly over the system.

Grid-based pixel block tracing divides the frame buffer into pixel blocks in a grid shape, and extracts the information on the whole screen with only a partial examination. An image or text screen is usually displayed on an LCD screen in a rectangular form, regardless of the use of a GUI environment. The image contents displayed in this rectangle belong to the same application, hence a minimal inspection on a relatively small area can predict display content. Figure 2 shows an example of the grid-based pixel block tracing. In Figure 2(a), FBMon divides the entire frame buffer into pixel blocks, and forms a group called *block partitions*, consisting of pixel blocks. Each block partition has a rectangular shape and consists of multiple pixel blocks. Let m represent the number of block partitions in each horizontal and vertical axis of LCD screen. Then, for an LCD screen of $X_{\text{res}} \times Y_{\text{res}}$ pixels, the number of pixels in a block partition is $(X_{\text{res}}/m) \times (Y_{\text{res}}/m)$, and the number of pixel blocks in one block partition is $\{X_{\text{res}}/(m \cdot k)\} \times \{Y_{\text{res}}/(m \cdot k)\}$. Here, the selection of a pixel block to represent the block partition is determined as follows. In Figure 2(b), let the coordinates of a pixel block in a block partition be (x, y), and let $a \times b$ be the block partition size. In the rth row of block partitions, FBMon reads pixel blocks of the same coordinates (x, y), and the coordinates for the x-direction increases on each row with a modular operation. That is, in the $(r+1)$th row of the block partition, FBMon reads the $(x+1 \mod b, y)$ block. Similarly, the pixel block read on the y direction changes after scanning the entire frame buffer. When the pixel block of (x, y) is read from the jth scan, then $(x, y+1 \mod a)$ is read in the $(j+1)$th scan. With this method—which examines only one pixel

DOI: 10.1002/spe

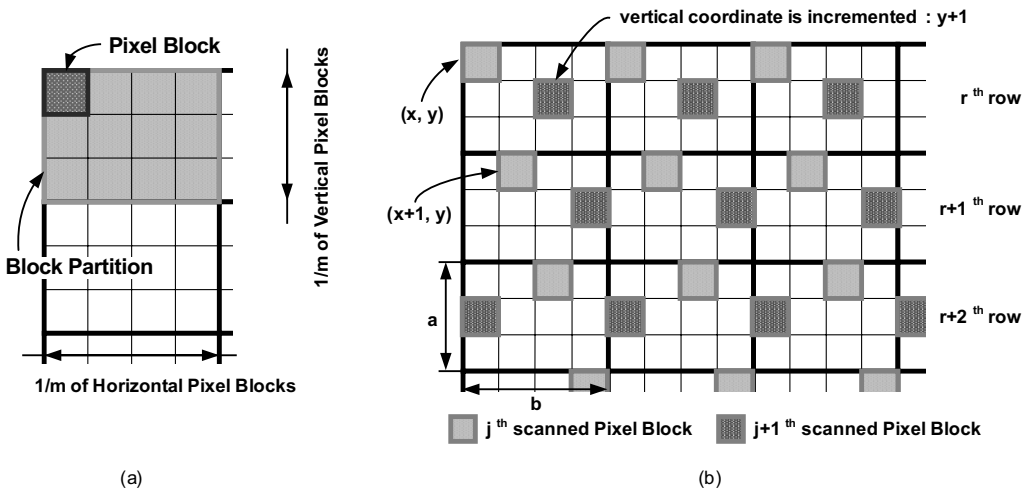


Figure 2. Grid-based pixel block tracing: (a) Frame buffer division; (b) *pixel block* search position for pixel block tracing.

block per block partition—the content type is determined with low overhead, and yet the overall screen is considered fairly. Meanwhile, *diagonal pixel reading* is used to classify a pixel block efficiently, into either image or text areas. This technique simply counts the number of pixels with different colors—according to the diagonals of the given pixel block—and determines the content type depending on its value. Hence, with a proper selection of pixel block width k and threshold t, an accurate classification of pixel blocks is possible using only $2k$ pixels located at diagonals, out of k^2 pixels in a pixel block.

Monitor load balancing is a technique to reduce the scanning overhead by dispersing the processing loads. This technique examines only one row of block partitions at every scan interval of p milliseconds and hence the amount of work is reduced $1/m$ compared with scanning the entire screen. The scanning interval p can be determined empirically by considering the actual overhead used to scan the whole screen in real systems. Since the time to scan the entire frame buffer and then to execute DRS is $T = p \times m$, the values of m and p are selected appropriately according to the user's QoS requirements.

In summary, FBMon effectively examines the frame buffer based on selective pixel readings, using grid-based pixel block tracing, diagonal pixel reading and monitor load balancing; it then determines the ratio of image area over the entire screen. The mechanisms minimize the overhead required to examine a whole frame buffer, and they collect information required by DRS. Equation (1) explains the number of pixels to be examined with the proposed mechanisms:

$$t \times m < \text{Pixels per } p \text{ ms} < 2k \times m \tag{1}$$

With a 640×480 pixel screen and $k = 16$, $t = 7$, $m = 10$ and $p = 10$, for example, only 70 pixels are examined every 10 ms with our mechanisms. From a practical point of view, the selection of parameter values is important; hence, extensive experiment results are presented in Section 3 regarding the optimal choice of the values.

DOI: 10.1002/spe

2.3. DRS

FBMon examines the frame buffer and classifies the current display contents into one of text area or image area, according to the ratio of image area to the entire screen. The key concept of DRS is to adjust the LCD refresh rate according to the display contents—that is, when text areas occupy most of the screen, the refresh rate can be reduced and hence power reduction can be achieved, because the display quality would not be degraded for this type of contents upon reducing the refresh rate. However, for the image area—which normally includes high-quality images or videos—the display quality should be maintained at a high level, and this forces fewer opportunities for reducing the refresh rate. A non-trivial case of applying DRS is frequently found in real applications; for example, the LCD screen for Web browsing typically consists of a mixture of both text and image areas. If the screen has a significantly higher portion of text than image, the refresh rate may stay low as users tend to concentrate more on text areas and not pay much attention to images. Similarly, displaying low-quality videos or images with a limited number of colors would not require high refresh rates, in most cases.

Inspired by the power reduction opportunities in displaying various types of content via refresh-rate control, we propose a simple refresh-rate control mechanism called DRS. The key idea behind it is that the refresh-rate of the display screen can be controlled appropriately by knowing the content type, via FBMon, and also by maintaining a 'hardware profile' for the underlying hardware. Here, the hardware profile contains *a priori* information on the flickering-free range of refresh rate for image-area ratios obtained by FBMon. The profile is prepared specifically for each hardware configuration because handheld devices may have different hardware characteristics, such as the LCD types (double-layer supertwist nematic (DSTN) or TFT), display resolutions and LCD controllers. DRS is executed at the FBMon renewal time; thus, the refresh rate can be controlled every p milliseconds, in the framework of monitor load balancing. At each interval, the display buffer is analyzed by FBMon and, depending on the ratio of image area to total screen area, the refresh rate is adjusted according to the information in the hardware profile.

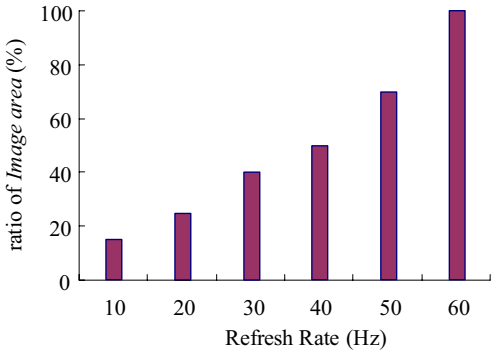
Figure 3 shows the hardware profile of an iPAQ3860 PDA. The graph on the left illustrates the experiment results regarding the relationship between the LCD refresh rate and the ratio of image area, which produces a flickering-free display quality. With 10 Hz of refresh rate, for instance, the ratio of image area over the entire screen should be lower than approximately 15%, in order not to produce a flickering display. With 60 Hz no flickering is experienced, even when the image area occupies 100% of the entire screen.

With this experiment, the range of allowable ratio for image area at each refresh rate can be obtained for use in the DRS operation. Figure 3(b) shows the hardware profile. When the ratio of image area is 0–15%, the refresh rate is set to 10 Hz; with a 15–25% ratio, the rate is adjusted to 20 Hz; and so on.

3. EXPERIMENTS

This section describes the experiment results of the proposed power-aware LCD management mechanism, and analyzes its performance characteristics.

DOI: 10.1002/spe



(a)

DRS State Description	Refresh Rate	Ratio of image area
	10 Hz	0 ~ 15 %
	20 Hz	15 ~ 25 %
	30 Hz	25 ~ 40 %
	40 Hz	40 ~ 50 %
	50 Hz	50 ~ 70 %
	60 Hz	70 ~ 100 %

(b)

Figure 3. Hardware profile for DRS operation (iPAQ3860): (a) flicker-occurring *image area* ratio at each refresh-rate (iPAQ 3860, TFT-LCD); (b) hardware profile of iPAQ 3860.

3.1. Experiment setup

The hardware platform used for the experiment is a Compaq iPAQ3860 PDA running the Linux kernel 2.4.19. The PDA is based on Intel's StrongARM (206 MHz SA1100) processor, and consists of 64 MB of SDRAM and 32 MB of flash memory. The display device is a 2.26×3.02 in^2 TFT-LCD providing a 320×240 resolution with 65 536 colors. The SA1100 processor is a highly integrated microprocessor that has a CPU core, memory and LCD controllers, and other peripheral controllers. The embedded LCD module of the SA1100 is used as the LCD controller in the iPAQ3860. The iPAQ platform used in our experiment provides five levels of backlight power control: super bright, high bright, medium bright, low bright and power save. We have used the fixed mode of 'low bright' in our experiment, which is the default value of backlight control utility supported by the Familiar GPE.

Both FBMon and DRS have been implemented by modifying the kernel. By adding codes to examine the memory space used by the frame buffer of the kernel, FBMon collects information on the contents of the current display. DRS is implemented by adding codes to control the LCCR0 register of the SA1100, to adjust the refresh rate. For the measurements of various system overheads, the operating system count register of SA1100—which runs with a 3.6864 MHz oscillator—is used. The GUI system used for the experiment is the Familiar GPE v0.7.2, which is one of the most common environments for handheld devices. The power consumption is measured over the whole PDA system, in order to show the efficiency of the proposed mechanism when applied to real systems. The Fluke multi-meter is used for the actual measurement of power.

3.2. Performance characteristics of FBMon

The first set of experiments aims to analyze the accuracy and performance of FBMon. Since these metrics are highly dependent on such configuration parameters as k, t, m and p, we have conducted a series of experiments with diverse combinations of parameters. The accuracy metric is to measure

DOI: 10.1002/spe

how correctly the proposed mechanism detects contents type, compared with the exhaustive method. The performance metric measures the response time and the system overhead of the proposed mechanism.

FBMon uses grid-based pixel block tracing and diagonal pixel reading, in order to reduce the monitoring cost of frame buffer. These approximation techniques should, however, not sacrifice the accuracy of the monitoring results. As FBMon examines only a single pixel block per every block partition of each scan, the accurate sampling of pixel block determines the overall accuracy. There are three parameters concerning the application of the proposed mechanisms: m, for the number of block partitions in each axis; k, for pixel block size; and t, for the threshold for image area. The combination of these parameter values produces different results with regards to accuracy and system overhead. In the following trials, we evaluate the accuracy of FBMon by experimenting with the system while using various combinations of m, k and t. Given the 320×240 screen resolution, parameters k and m are appropriately selected, and t varies within the range of k. The number of pixels to be examined in each scan is calculated by Equation (2):

$$t \times m^2 < \text{Number of pixels per frame buffer} < 2k \times m^2 \tag{2}$$

The FBMon accuracy is evaluated by an error in obtaining the ratio of image area via FBMon, compared with the ratio obtained by the exhaustive scanning of the entire screen. Figure 4 shows the sample snapshots used for the experiments. Apart from basic image samples, four GUI-based application programs—*gpe-gallery* for image viewer, *gpe-sketchbook* for simple drawings, *gpe-edit* for text editing, and *gpe-calendar* for text-based schedule management—are used to generate various kinds of screen snapshots that consist of both text area and image area of different quality.

Table II shows both the monitoring accuracy and the estimated system overhead, only for cases where the average error is approximately less than 12% and the standard deviation is less than 10. These are the thresholds that would dictate the usefulness of the proposed mechanism. Here, the system overhead is estimated by the range of maximum and minimum number of pixels to be examined (Equation (2)). Among the cases listed in the table, $k = 20$, $m = 4$, $t = 7$ is considered to produce the best result in terms of both accuracy and system overhead. With this parameter setup, FBMon can monitor the frame buffer by examining a maximum of 640 pixels, out of 320×240 pixels, only to result in approximately 6.5% of average error, with a little variation. This result validates the effectiveness of the proposed method—that is, with less than 1% of pixel examination, FBMon produces a monitoring accuracy of less than 7% when compared with the time-consuming exhaustive method.

As described in Section 2.2, FBMon uses a monitor load-balancing technique to reduce the system overhead, by examining only one row of block partitions at every p milliseconds. Depending on the value of p, therefore, users may experience a response time delay. The following experiment aims to understand the relationship between the processor overhead and the response delay. The processor overhead O_p is defined as the ratio of actual processor time t_1 spent in running FBMon over p milliseconds: $O_\mathrm{p} = t_1/p$. The response delay T_r is measured by the time delay required for a change in display content to be detected by FBMon. In order for a change in screen content to be detected successfully, an entire scan of the frame buffer is necessary; the overall response delay is calculated as $T_\mathrm{r} = p \times m$, because FBMon examines only one row of block partitions at every p milliseconds.

Table III shows the measured values of processor overhead and response delay with different scanning intervals p, for the case of $k = 20$, $m = 4$, $t = 7$, which produced the best accuracy. In our experiments, p varies between 10 and 50 ms in intervals of 10 ms. The experiment results show that as

DOI: 10.1002/spe

Figure 4. Sample images used for the evaluation (320 × 240, 16 bpp).

Table II. Monitoring accuracy and overhead estimation according to k, m and t.

			Monitoring accuracy		Overhead estimation	
k	m	t	Average error (%)	Standard deviation	Minimum (pixels)	Maximum (pixels)
5	2	4	10.0	9.2	16	40
	8	4	11.2	9.1	256	640
8	2	5	7.8	8.9	20	64
	5	4	8.9	8.2	100	400
	10	4	8.7	8.0	400	1600
10	2	5	6.6	6.8	20	80
	4	5	6.5	8.1	80	320
	8	5	7.9	7.9	320	1280
16	5	7	6.7	8.1	175	800
20	2	7	7.3	4.2	28	160
	4	7	6.5	4.3	128	640

DOI: 10.1002/spe

Table III. Overhead and response time characteristics.

K	m	t	p (ms)	Overhead (t_1/p, %)	Response time ($Tr = p \times m$, ms)
20	4	7	10	0.025	40
			20	0.012	80
			30	0.008	120
			40	0.006	160
			50	0.005	200

p increases, the response delay increases proportionally, but the processor overhead decreases with relatively small variation. In fact, with the SA1100 processor, the overhead of running FBMon is measured as being very small—approximately 0.025% in its worst case—and in this case, the response delay is less than 50 ms, which is usually considered the maximum tolerable delay experienced by a human being [18]. This result indeed validates the proposed mechanism's efficiency in terms of monitoring overhead and response delay, and hence it is a practical mechanism easily implemented on real systems.

3.3. Power reduction effects of DRS

Having understood the performance characteristics of various parameters required for FBMon operations, the second set of experiments is conducted to analyze the power-saving effects of the proposed DRS scheme. The experiments are set up to run various kinds of actual applications on a PDA, the iPAQ3860 and to measure the refresh-rate distribution as well as the amount of energy saved via DRS. Three popular applications are used for the experiments: *gpe-edit*, *gpe-gallery* and *mpegplay*. The PDA-based text editor *gpe-edit* is used to benchmark a typical of user-interactive text-editing scenario, and the energy consumption is measured during the session. For the image applications, the *gpe-gallery* imager viewer is used, and the screen snapshots shown in Figure 4 are slide-showed at intervals of 5 seconds. As for the video applications, the *mpegplay* video decoder is used to play back three different kinds of 120 kbps compressed video clips, with 5-second intervals between the clips. The experiments are run under the configuration of $k = 20$, $m = 4$, $t = 7$, $p = 10$ to provide the best results with FBMon.

Figure 5 shows the refresh-rate distributions for each type of application running. For *gpe-edit*, DRS reduces the refresh rate to 10 Hz without degradation of display quality. In the case of image viewing (*gpe-gallery*) and video playback (*mpegplay*), however, DRS selects a wide range of refresh rates at runtime and the portion of high refresh rates is noticeably larger than is the case with simple text-based applications. This shows that DRS works as expected, and the display QoS is maintained.

The actual power consumption has been measured to understand the performance of DRS. Table IV shows the measured power consumptions, in terms of energy spent in unit time (i.e. Watts), for three types of application and when running on two different system configurations: a system with no refresh-rate control policy (i.e. the refresh rate is fixed at 60 Hz) and the proposed FBMon–DRS system.

DOI: 10.1002/spe

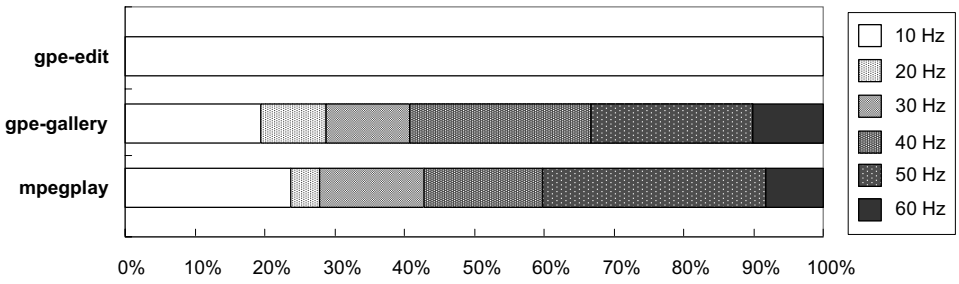


Figure 5. Refresh-rate distributions.

Table IV. Power consumption.

	No policy (Watts)	PBMon–DRS (Watts)	Energy gain
gpe-edit	1.26	1.13	10.3%
gpe-gallery	1.21	1.16	4.1%
mpegplay	1.59	1.52	4.4%

It is apparent from the experiment results that in the case of text-editing, the energy-reduction effect of DRS is good—approximately 10%—due to the scaled refresh rate of 10 Hz. This result is, in fact, considered significant because the battery lifetime of handheld devices can be practically extended with a non-trivial amount of time, even with a simple technique such as the refresh-rate control. In the case of image viewing and video playback, the energy reduction via DRS is relatively small, approximately 4%. The refresh-rate distribution shown in Figure 5 explains the results. Note that the average energy consumption shown in Table IV is obtained with different experiment scenario for each application type: the energy consumption for the text-editing application, for example, is measured for a given time interval of user interaction, whereas the measurements in image viewing and video playback are conducted, based on the slide-show scenario with a certain idle-time interval. Therefore, in our experiments, it is meaningless to compare the energy consumption of different applications directly.

4. CONCLUSIONS

In this paper, we present a power-aware LCD management mechanism based on the runtime frame buffer monitoring FBMon, and a dynamic refresh-rate scaling DRS. FBMon classifies the content type displayed in the LCD screen with an accuracy greater than 93%, and with an implementation overhead less than 0.01%. DRS is empirically validated on a commercial PDA with a 320×240 resolution,

DOI: 10.1002/spe

and the energy gain is measured at between 4% and 10%, depending on the end-user application, without a degradation of the display content quantity. Although the proposed mechanism is implemented and experimented currently on a 320×240 screen resolution, a significant power reduction can be expected with high-resolution LCDs. One study [5] reports that with a 640×480/16 bpp/60 Hz display system, a maximum power reduction of about 27% can be obtained by scaling the refresh rate to 30 Hz. This is mainly due to the reduction in the amount of data transfer (147 Mbps, in this case) between the frame buffer and LCD controller. Considering the recent hardware advances in handheld devices, the LCD resolution tends to increase and therefore the effect of DRS becomes more significant in terms of power saving in such devices. Finally, as FBMon and DRS are developed entirely within a software framework, the techniques can be easily applied to any embedded operating system without modifications at the application level.

The dynamic refresh-rate control mechanism presented in this paper is only a part of the overall efforts to reduce LCD-related power consumption in handheld devices. As discussed at the outset of this paper, there are other known techniques—such as backlight or color depth control—to be used together with DRS. We believe that the frame buffer monitoring technique can be easily applied to the development of effective software mechanisms that use various kinds of LCD power management. This is part of our future work.

ACKNOWLEDGEMENTS

This work was supported in part by the National Research Laboratory (NRL) program of the Korea Science and Engineering Foundation (2005-01352), the ITRC programs (MMRC, HY-SDR) of IITA, and the Seoul R&BD Program of City of Seoul, Korea.

REFERENCES

1. Lu Y-H, Benini L, De Micheli G. Power-aware operating systems for interactive systems. *IEEE Transactions on Very Large Scale Integration Systems* 2002; **10**(1):119–134.
2. Govil K, Chan E, Wasserman H. Comparing algorithms for dynamic speed-setting of a low-power CPU. *Proceedings of ACM International Conference on Mobile Computing and Networking*, November 1995. ACM Press: New York, 1995; 13–25.
3. Udani S, Smith J. The Power Broker: Intelligent power management for mobile computers. *Technical Report MS-CIS-96-12*, Distributed Systems Laboratory, Department of Computer Information Science, University of Pennsylvania, 1996.
4. Choi I, Shim H, Chang N. Low-power color TFT LCD display for hand-held embedded systems. *Proceedings of the International Symposium on Low Power Electronics and Devices*, August 2002. ACM Press: New York, 2002; 112–117.
5. Gatti F, Acquaviva A, Benini L, Ricco' B. Low power control techniques for TFT LCD displays. *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, October 2002. ACM Press: New York, 2002; 218–224.
6. Cheng W-C, Liang J-L, Pedram M. Software-only bus encoding techniques for an embedded system. *Proceedings of ASP-DAC/VLSI Design*, January 2002. IEEE Computer Society Press: Washington, DC, 2002; 126–131.
7. Zhong L, Jha NK. Graphical user interface energy characterization for handheld computers. *Proceedings of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems*. ACM Press: New York, 2003; 232–242.
8. Flinn J, Satyanarayanan M. Energy-aware adaptation for mobile applications. *Proceedings of the Symposium on Operating Systems Principles*, December 1999. ACM Press: New York, 1999; 48–63.
9. Dalton AB, Ellis CS. Sensing user intention and context for energy management. *Proceedings of the Workshop on Hot Topics in Operating Systems (HotOS)*, January 2003. USENIX Association: Berkeley, CA, 2003; 151–156.

DOI: 10.1002/spe

10. Iyer S, Luo L, Mayo R, Ranganathan P. Energy-adaptive display system designs for future mobile environments. *Proceedings of the 1st International Conference on Mobile Systems, Applications, and Services (MobiSys 2003)*, May 2003. USENIX Association: Berkeley, CA, 2003; 245–258.
11. Harter T, Vroegindeweij S, Geelhoed E, Manahan M, Ranganathan P. Energy-aware user interfaces: An evaluation of user acceptance. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, April 2004. ACM Press: New York, 2004; 199–206.
12. Bloom L, Eardley R, Geelhoed E, Manahan M, Ranganathan P. Investigating the relationship between battery life and user acceptance of dynamic energy-aware interfaces on handhelds. *Proceedings of the 6th International Symposium on Mobile Human–Computer Interaction (MobileHCI)*, September 2004 (*Lecture Notes in Computer Science*, vol. 3160). Springer: Berlin, 2004; 13–24.
13. Kamijoh N, Inoue T, Michael Olsen C, Raghunath MT, Narayanaswami C. Energy trade-offs in the IBM wristwatch computer. *Proceedings of the 5th International Symposium on Wearable Computers*, Zurich, Switzerland, October 2001. IEEE Computer Society Press: Washington, DC, 2001; 133–140.
14. Narayanaswami C, Raghunath MT. Application design for a smart watch with a high resolution display. *Proceedings of the International Symposium on Wearable Computing*, Atlanta, GA, October 2000. IEEE Computer Society Press: Washington, DC, 2000; 7–14.
15. Shim H, Chang N, Pedram M. A backlight power management framework for battery-operated multimedia systems. *IEEE Design and Test of Computers* 2004; **21**(5):388–396.
16. Cheng W-C, Hou Y, Pedram M. Power minimization in a backlit TFT-LCD display by concurrent brightness and contrast scaling. *IEEE Transactions on Consumer Electronics* 2004; **50**(1):25–32.
17. Brakmo LS, Wallach DA, Viredaz MA. μ Sleep: A technique for reducing energy consumption in handheld devices. *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services (MobiSys 2004)*, Boston, MA, June 2004. ACM Press: New York, 2004; 12–22.
18. Shneiderman B. *Designing the User Interface: Strategies for Effective Human–Computer Interaction*. Addison-Wesley: Reading, MA, 1998.

DOI: 10.1002/spe