Content-Adaptive Display Power Saving in Internet Mobile Streaming

Yao Liu[1], Mengbai Xiao[2], Ming Zhang[2], Xin Li[3], Mian Dong[4], Zhan Ma[5], Zhenhua Li[6], Songqing Chen[2]

[1]SUNY Binghamton
yaoliu@cs.binghamton.edu

[2]George Mason University
{mxiao3, mzhang8,sqchen}@gmu.edu

[3]Samsung Telecommunications America
x.li@samsung.com

[4]AT International, Inc.
dongmian@gmail.com

[5]Nanjing University
zhan.ma@gmail.com

[6]Tsinghua University
lizhenhua1983@tsinghua.edu.cn

ABSTRACT

Backlight scaling is a technique proposed to reduce the display panel power consumption by strategically dimming the backlight. However, for Internet streaming to mobile devices, a computationally intensive luminance compensation step must be performed in combination with backlight scaling to maintain the perceived appearance of video frames. This step, if done by the CPU, could easily offset the power savings via backlight dimming. Furthermore, computing the backlight scaling values requires per-frame luminance information, which is typically too energy intensive to compute on mobile devices.

In this paper, we propose Content-Adaptive Display (CAD) for Internet mobile streaming. CAD uses the mobile device's GPU rather than the CPU to perform luminance compensation at reduced power consumption. Backlight scaling schedule is computed using a more efficient dynamic programming algorithm than existing work. We implement CAD within an Android app and use a Monsoon power meter to measure the real power consumption. Experiments are conducted on more than 470 randomly selected YouTube videos, and results show that CAD can effectively produce power savings.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications

General Terms

Algorithm, Experimentation, Measurement

Keywords

Internet Mobile Streaming, LCD, Display, Power Saving, Backlight Scaling

NOSSDAV'15, March 18-20, 2015, Portland, OR, USA
Copyright 2015 ACM 978-1-4503-3352-8/15/03 $15.00
http://dx.doi.org/10.1145/2736084.2736087.

1. INTRODUCTION

Video streaming on mobile devices is limited by the battery capacity. According to Carroll and Heiser, the display subsystem is responsible for about 38% to 68% of the total power consumption during video playback [4]. However, unlike the wireless network interface cards (WNICs), whose power consumption can be reduced by putting them into low power sleep mode for as long as possible, mobile display panels cannot be put into sleep mode and must be kept active throughout the video playback.

To save power consumed by Liquid-Crystal Displays (LCD), backlight scaling has been proposed. This technique reduces power consumption by dimming the display backlight. Meanwhile, the brightness perceived by the human eye is maintained by increasing the affected image's luminance. By simultaneously scaling the backlight and increasing the luminance of the image, the original image can be rendered with little distortion.

However, implementing the backlight scaling strategy with luminance compensation to save power during *video playback* is challenging. *First*, backlight scaling values must be determined subject to the following constraints: (i) To maintain image fidelity, the backlight level cannot be lower than a point determined by the brightness characteristics of an image. This constraint requires that the maximum pixel luminance of every frame be computed, which can be both time and energy intensive; (ii) It is infeasible to adjust the backlight level for every frame because the display hardware takes time to perform the adjustment; and (iii) Large inter-frame backlight variation can cause flickering effects, constraining the range of such adjustments. *Second,* luminance compensation must be performed for every pixel in every video frame. Thus, increasing the luminance for the entire frame of a high resolution video on a high resolution display could consume tens of millions of CPU cycles. While a powerful CPU could complete this task in real time, the corresponding power consumption overhead could offset the power saved by dimming the backlight. Therefore, previous studies often suggested these tasks should be performed offline using extra computing resources [6, 12], making backlight scaling hardly practical.

In this paper, we propose a Content-Adaptive Display (CAD) power saving mechanism for reducing display power consumption during Internet mobile streaming sessions. Instead of using the CPU, CAD uses the OpenGL ES API to interface with the Graphics Processing Unit (GPU) on mobile devices to adjust pixel luminance during video streaming playback, allowing a net power savings in combination with the backlight scaling strategy. Furthermore, CAD employs a dynamic programming approach for determining backlight scaling assignments, which has a lower complexity than existing algorithms. The per-frame luminance information required by the dynamic programming algorithm is computed offline using external computing resources.

To evaluate the effectiveness of CAD, we implement it within a mobile video player application (app) on the Android platform. During video playback, the mobile app sets the backlight according to the backlight scaling assignment and simultaneously compensates the brightness by increasing pixel luminance through GPU computations. We randomly select more than 470 YouTube videos to evaluate CAD. A Monsoon power meter is used for real measurements of power consumption. Results show that with CAD, our video player app can save power for 67.3% videos with negligible (up to 5%) pixel distortion, and 83.4% videos if 10% pixel distortion is allowed. For more than 44% of the videos in our study, CAD can achieve more than 500 mW of power savings.

The remainder of the paper is organized as follows. Section 2 discusses background and some related work. We present the design of our Content-Adaptive Display power saving mechanism in Section 3. The implementation is described in Section 4, and the evaluation is discussed in Section 5. We make concluding remarks in Section 6.

2. BACKGROUND AND RELATED WORK

Previous studies have pointed out that the backlight of an LCD display dominates the energy consumption of the display subsystem [4, 14]. Therefore, power can be saved if we reduce the backlight brightness level of the LCD display. However, simply dimming the backlight can lead to image distortion, which is normally defined as the resemblance between the original video image and the backlight-scaled image [6, 15]. One way to resolve the problem is by simultaneously scaling the backlight level and increasing the luminance of every pixel [5, 7, 9, 12]. In this way, image fidelity can be preserved.

While the idea of increasing the luminance of video frames at runtime to compensate for a dimmed backlight is intuitive, a significant challenge arises when attempting to adjust the luminance of every pixel in every video frame in an energy efficient manner. Many existing techniques rely on the CPU to perform per-pixel manipulation [7, 9] but this may significantly offset the power savings achieved via backlight dimming. To avoid the computation intensive step on the mobile device, Hsiu et al. and Lin et al. propose not to perform luminance compensation, but instead, simply to choose a critical backlight level for each frame as the scaling constraint [10, 11]. This, however, can lead to a large amount of distortion of the displayed frames. Instead of using the CPU, Ruggiero et al. consider a special multimedia processor, the Freescale i.MX31, which has an Image Process Unit (IPU) that can be used to perform the luminance adjustment task [13]. Rather than a client-side solution, Pasricha et al. [12] and Cheng et al. [6] propose to migrate the computation to an intermediate proxy server that computes backlight scaling values and transcodes the original video to a luminance-adjusted version.

Compared to existing work, our proposed solution can save display consumption while maintaining image fidelity without relying on either a specialized processor for pixel luminance compensation or intermediate servers for video transcoding. Instead, CAD uses the Graphics Processing Unit (GPU) to perform the computation.

3. DESIGN OF THE CONTENT-ADAPTIVE DISPLAY (CAD) POWER SAVING MECHANISM

In this section, we discuss how CAD generates backlight scaling data offline using a dynamic programming algorithm with lower complexity and how CAD performs luminance compensation online using GPU instead of CPU.

3.1 Display Power Saving Constraints

CAD is based on the *Backlight Scaling* method [9], which saves power consumption by dimming the backlight level. Suppose the display brightness levels lie within the range $(0, 1]$, where a value of 0 indicates that the backlight is off, and a value of 1 indicates that it is set to maximum brightness. With backlight scaling, for every frame in the video, we would like to set the display backlight level to $b \in (0, 1]$. To avoid fidelity loss under reduced backlight levels, we adopt *Luminance Compensation* [6, 15]. We increase the luminance of every pixel in the frame by a factor of $\frac{1}{b}$ (i.e., increasing the Y component of the YUV representation of every pixel). In this way, the *observed* pixel luminance, Y', is adjusted to a value that is the same as that of the original frame's: $b \times Y' = b \times Y \times \frac{1}{b} = Y$. With joint backlight scaling and luminance compensation, display power consumption can be saved without any observable fidelity loss.

For every frame in a video, the backlight scaling value must be determined according to the following three constraints:

Distortion Constraint. The Y component a pixel can not be scaled to higher than its maximum value, 255. Therefore, if b is chosen to be a value such that $Y \times \frac{1}{b} > 255$, then the observed luminance of the adjusted pixel with the reduced backlight will be lower than the luminance under the original brightness level, distorting the displayed image. In previous work this distortion has been referred to as a “clipping artifact” [8]. To avoid creating these “clipping artifacts”, we must limit any backlight adjustments to $b \geq \frac{Y_{max}}{255}$, where Y_{max} is the maximum pixel luminance in the frame. This distortion-based constraint gives rise to a *lower bound* on the adjusted backlight level for every frame in the video.

User Experience Constraint. While setting the backlight level of every frame to its lowest possible value subject to distortion constraint can maximize power savings, users could experience inter-frame brightness distortion, often perceived as flickering, if the variation in backlight scaling levels between two consecutive frames is too big. Therefore, we need to *limit* the brightness variation between two consecutive frames to reduce this flickering effect.

Hardware Constraint. The display hardware requires a minimum amount of time to apply any brightness adjustments. Therefore, it is impossible to adjust the backlight level promptly for every video frame. Instead, we must specify a *minimum interval* (in terms of numbers of frames) where the backlight level must remain constant for all frames.

3.2 Computing Backlight Scaling

Given the constraints described above, we propose an offline algorithm that uses global knowledge about all frames in the video for computing the backlight scaling level for maximized power saving.

We have designed a dynamic programming algorithm to compute backlight scaling levels given a sequence of maximum pixel luminance values of video frames as well as a set of constraints to maintain an acceptable video-viewing experience. As discussed earlier, two constraints are necessary. The first constraint, ℓ_{min}, specifies the minimum interval in while backlight must remain at the same level for all frames. The second constraint, specified by Δ_b, limits the ratio of change in the backlight level. That is, if the backlight is at level b_t at time t and b_{t+1} at time $t+1$, then we enforce the constraint $b_t \times (1 - \Delta_b) \leq b_{t+1} \leq b_t \times (1 + \Delta_b)$.

Our dynamic programming algorithm computes the values of $B(t, b)$, the minimum cumulative backlight levels ending at frame t with the backlight level of the final interval set to b. $B(t, b)$ can be computed by the following recurrence:

$$B(t, b) = \min_{t',b'}(b \times (t - t') + B(t', b')) \qquad (1)$$

where t' is the frame number of the end of the interval that immediately precedes the interval frame t belongs to, subject to the constraint $t - \ell_{\max} \leq t' \leq t - \ell_{min}$ Here, b' is the brightness level of the previous interval, and also subjects to the Δ_b constraint discussed above. This algorithm will minimize power consumption if a linear relationship exists between backlight levels and display power. We confirm that such a linear relationship exists in Figure 3.

We also add an additional constraint, ℓ_{max}, specifying the maximum length of an interval. This decision is motivated by the two considerations. First, setting the algorithm to use fixed-length intervals of constant brightness produces suboptimal behavior. This is because large changes in maximum luminance exist at many positions within a video. These changepoints are unlikely to align with boundaries of any preset fixed-length interval. As a result, a large number of fixed-length intervals would cross changepoint boundaries, leaving a portion of these intervals assigned to higher backlight levels than necessary. Therefore, to achieve near-optimal brightness levels we must consider intervals of varying lengths. On the other hand, if we were to consider all possible lengths of constant-brightness intervals, the algorithm would be optimal but computation would have a complexity of $O(T^2 \times |b|^2)$[1], which is unsuitable for our application. Motivated by the fact that long intervals of constant brightness can be expressed by concatenating shorter intervals, we can choose constant brightness intervals whose length is not fixed, but lie within a small range of values (i.e., between ℓ_{min} and ℓ_{max}). This allows regions of constant brightness intervals to align more closely with a video's luminance profile, thus achieving near-optimal total brightness levels. Therefore, the ℓ_{max} constraint can reduce the complexity of our algorithm at the cost of only a minimal increase in total brightness over the course of video playback.

[1]This is similar to the algorithm proposed by Lin et al. for determining optimal backlight scaling levels with $O(N^2M^2(N + \ln M + d^2))$ complexity, where N is the number of frames, M is the number of backlight levels, and d is the minimum duration before a backlight level change [11].

```
1: ▷ On input $lum$, an array of length $T$, indicating maximum luminance values for each frame of a video containing a total of $T$ frames, compute $output$, an array containing minimum backlight brightness values subject to constraints.
2: ▷ Compute the recurrence
3: for $t$ in $[\ell_{min}, T]$ do
4:     for $t'$ in $[t - \ell_{max}, t - \ell_{min}]$ do
5:         for $b \geq \max(lum[t' + 1 : t])/255$ do
6:             for $b'$ in $[b/(1 + \Delta_b), b/(1 - \Delta_b)]$ do
7:                 if $b \times (t - t') + B[t', b'] < B[t, b]$ then
8:                     $B[t, b] = b \times (t - t') + B[t', b']$
9:                     $H[t, b] = (t', b')$
10: ▷ Backtracking phase
11: output = array(T)
12: $t = T$
13: $b = \arg\min_{b'} B[t, b']$
14: while $t >= 0$ do
15:     $(t', b') = H[t, b]$
16:     output$[t' + 1 : t] = b$
17: return output
```

Figure 1: Computing the backlight scaling data with $O(T \times (\ell_{max} - \ell_{min} + 1) \times |b|^2)$ complexity, where $|b|$ indicates the number of possible values of b. In the algorithm, $B[t, b]$ indicates the sum of the brightness levels ending at frame b whose final constant-brightness interval level is b, T indicates the number of frames in the video, ℓ_{min} is the shortest allowed constant-brightness interval (in frames) and ℓ_{max} is the longest, $lum[t' : t]$ indicates the luminance values over video frames t' through t, Δ_b encodes the constraint specifying the allowable ratios of adjacent brightness intervals, and $H[t, b]$ is a history array that records how the minimum-brightness array $B[t, b]$ was constructed.

Our algorithm thus consists of a forward step where the values of $B(t, b)$ are computed and a backward step where the values of b_t, the backlight value at video frame, t, are recovered. Pseudocode to compute the dynamic programming recurrence is shown in Figure 1.

3.3 Backlight Scaling and Luminance Compensation

A mobile device can use the backlight scaling information computed offline to dynamically adjust its LCD display brightness level during playback. Meanwhile, to compensate for the dimmed display and maintain image fidelity, the mobile device increases the luminance level for every pixel in every frame that is associated with a dimmed backlight brightness level. That is, given the backlight scaling level b_f for rendering frame f, we increase the luminance of every pixel p in frame f from Y_p, its original luminance, to $Y'_p = Y_p \times \frac{1}{b_f}$, and retain the original U_p and V_p values. This luminance scaling allows the video frame to be rendered on the display without fidelity loss. Note that this luminance compensation has to be performed on all frames that are rendered at scaled backlight level to maintain consistent contrast levels. This computation must occur as long as the backlight intensity is not set to its default value.

This luminance compensation step requires a floating point data operation for every pixel in every frame. Therefore, the total computation load generated by operations on an entire frame would be infeasible for the CPU to perform given the time constraints of live video playback. In addition, even if CPUs operating at higher frequency could accomplish this task in time, the extra power consumed by the CPU could offset the savings achieved by dimming the backlight. The GPU, on the other hand, is able to perform a large number of tasks in parallel, enabling it to compute adjusted luminance over many pixels in a timely manner. Therefore, we use the OpenGL ES API to interface with mobile device GPUs, enabling these GPUs to perform the luminance compensation task in an online manner.

3.4 Deciding Whether to Use Display Adaptation

While using the GPU for luminance compensation, we need to take into account the additional power consumed by the GPU, i.e., we need to determine whether net power savings can be achieved by comparing GPU incurred power consumption overhead with display power that can be saved via backlight scaling. Our approach toward this problem involves three steps: (i) We build models to estimate the power consumption of different devices at different backlight brightness settings during video playback using power measurement results. We also estimate a fixed rate of power consumption of the GPU on these devices based on power measurement results. (ii) We use these models, given the input of a sequence of minimum backlight scaling values calculated using the dynamic programming algorithm, to estimate the display energy consumption during video playback. (iii) We compare these energy consumption values with baseline values to estimate the total energy saved, then compare this saved amount to our estimate of GPU energy consumption. If the value of display energy savings exceeds that of GPU energy consumption, then we can be reasonably confident that backlight scaling will save power when playing the video, and we can apply our scaling method. Although savings may not be possible for all videos, over the course of typical mobile device usage, significant savings could be achieved.

4. IMPLEMENTATION

4.1 Generating Backlight Scaling Data

Backlight scaling schedule is computed externally by a standalone application that attempts to maximize power savings. The external application first decodes the video using the ffmpeg library [1]. Then, it determines the minimum backlight level required using the dynamic programming algorithm described in Figure 1. We also compute backlight scaling information that results in greater power savings but causes a small number of pixels to be displayed with an observed luminance lower than the original luminance. We refer to this as "pixel distortion". If up to $d\%$ pixel distortion per frame can be tolerated, we can choose the $(100 - d)$ percentile luminance of every frame instead of the maximum luminance as input to the dynamic programming algorithm. The generated backlight scaling data is stored in a file with each value coupled with the corresponding frame index. This data is then sent to the mobile video player app before video playback.

4.2 Runtime Backlight Scaling and Luminance Compensation

We implement the concurrent backlight scaling and luminance compensation in an Android video player app. To program GPU to increase pixel luminance, we use OpenGL for Embedded Systems (OpenGL ES) [3]. OpenGL ES is a subset of the OpenGL 3D graphics API. It is designed for handheld and embedded devices such as mobile phones, PDAs, and video game consoles. Notable platforms supporting OpenGL ES 2.0 include iPhone 3GS and later versions, Android 2.2 and later versions, and WebGL.

In the video player app, we use the `MediaPlayer` provided by Android to decode the video stream. Instead of rendering the decoded video frames onto the default `Surface`, we create a `GLSurfaceView`, wrap it into a `Surface` object, and set the `MediaPlayer` to use this `Surface` as the video data sink so as to divert the video frames into the `GLSurfaceView` object.

To adjust the luminance of all pixels of the frame as a result of an adjusted brightness level, we set up a customized `Renderer` in the `GLSurfaceView` that implements the Vertex Shader and the Fragment Shader. Since pixels have been converted by the Android system from YUV to RGB color space for rendering, the Fragment Shader must to convert the color space back to YUV before luminance compensation can be performed. The Vertex Shader just sets up the vertex positions without any transformation. Next, we scale the Y value of the pixel to Y' and convert the YUV representation back to RGB color space using Y', U and V. By scaling the backlight brightness level and the pixel luminance simultaneously, power consumption is reduced and the image fidelity is maintained.

5. EVALUATION

To evaluate our CAD during video streaming playback, we have installed our video player app on various Android devices. Due to space limit, in this paper, we only show results from a Samsung Galaxy Tab 2 10.1-inch tablet. It features the TI OMAP4430 SoC which includes the PowerVR SGX540 GPU, programmable using OpenGL ES 2.0. The LCD displays on this device support 255 backlight levels. To build the supported backlight level table, we evenly pick 255 numbers in the range $(0, 1]$.

We uniformly select 471 YouTube videos at random for evaluation using random prefix sampling method proposed by Zhou et al. [16]. We download the 360P version of all 471 videos for experiments and analysis. Figure 2 shows the setup of our experiments. To accurately measure the power consumption during video playback, we use a Monsoon power monitor [2] to supply power directly.

5.1 Display and GPU Power Consumption Models

We build a display power consumption model as a function of brightness. We play a video with the "Gallery" app supplied by Android and measure its power consumption using the Monsoon power monitor. To maintain reasonable user experience, we restrict the minimum normalized backlight luminance level to 0.5. The results are shown in Figure

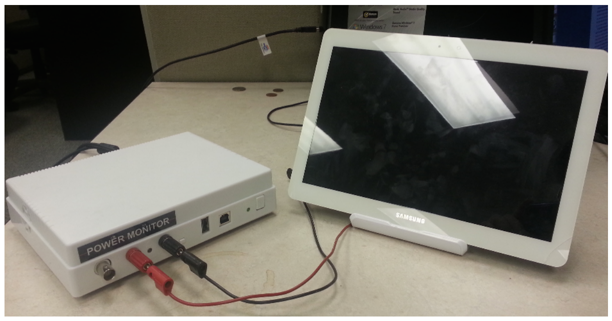

Figure 2: Power measurement setup.

3. We find that the display power consumption of 10.1-inch Galaxy Tab can be best represented with the following linear model: $y = w_1 \times b + w_2$, where $b \in [0.5, 1]$ is the normalized display backlight level, $w_1 = 3512.7$, and $w_2 = 1053.4$, with $R^2 = 0.9928$. In addition, we also measure the GPU power consumption on the 10.1-inch tablet. We set the backlight to maximum level and compare power consumption when GPU is not used with when GPU is used to scale pixel luminance by 1.0 (no effect). Power measurement results show that when using GPU for luminance compensation to play videos at 30 frames per second, the GPU consumes a constant amount of power, around 578 mW.

We use our display and GPU power models in combination with computed backlight scaling data to decide whether the CAD mechanism should be employed to save power.

5.2 Power Saving with Backlight Scaling

To calculate backlight scaling schedule, we first determine appropriate values for parameters required in the algorithm, including ℓ_{min} and Δ_b. We set these parameters to different values and compute the corresponding backlight scaling. Five users were asked to watch videos played using our CAD mechanism and report whether they noticed flickering or distortion during playback. We found that when we set Δ_b to 0.06, users did not perceive any flickering. We thus use 0.06 for backlight scaling data generation. Similarly, we found that setting $\ell_{min} = 5$ produces no observable display hiccups. We therefore enforce that the display backlight level remains stable for at least 5 frames.

We run the dynamic programming algorithm on all 471 YouTube videos. For each video, we decode the video and extract the maximum, 98-percentile, 95-percentile, and 90-percentile pixel luminance for every frame. We use this data as input to our dynamic programming algorithm, which computes the backlight scaling level assignment for each frame that will yield no distortion, up to 2%, 5%, and 10% pixel distortion per frame respectively. Given the backlight scaling level assignment data, we further use the power consumption models to examine if and how much net power saving can be achieved (i.e., display power saving being greater than GPU power consumption). Figure 5 shows the results. In this figure, "schedule-0%" represents the setting where backlight scaling data is computed offline using maximum luminance data per frame, while "schedule-2%" represents the experiment setting where backlight scaling data is computed offline using 98-percentile luminance data per frame. Results show that if no distortion is allowed, 17 out of 471 videos can save power with the CAD mechanism. If more distortion can be tolerated, more power can be saved: 317 (67.3%) videos can save power with negligible (up to 5%) pixel distortion, If up to 10% pixel distortion is allowed, 393 (83.4%) videos can save power, and 209 (44.4%) videos can save more than 500 mW on average. For backlight scaling schedules that may produce pixel distortion, we calculate their peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) between the rendered video and the non-backlight-scaled video. Figure 4 shows that for all videos, their PSNR values are always above 40 dB when up to 2%, 5%, or 10% pixels are rendered with lower observed luminance than their original luminance, indicating good rendered frame quality. The SSIM values between the rendered videos and the non-backlight-scaled videos are always more than 0.99999.

We also use the power monitor to measure **real** power savings for videos that are expected to save power via our CAD mechanism. Figure 6 shows the overall power consumption of playing one video under different settings. This video is 9 minute 56 seconds long (596 seconds) and is encoded at 30 frames per second. For the 10.1-inch tablet, when our CAD mechanism is not used, GPU is put into sleep mode, the average overall power consumption is 4898 mW. When we apply backlight scaling that yields no distortion, the average overall power consumption is 4963 mW, slightly higher than without adaptation. On the other hand, if up to 2% pixel distortion is allowed, the average power consumption can be reduced to 3518 mW, a 28% savings. If up to 5% pixel distortion is allowed, the average power consumption is further reduced to 3264 mW, a 33.4% savings.

6. CONCLUSION

In this work, we have designed and implemented a Content-Adaptive Display (CAD) power saving mechanism for reducing display power consumption on mobile devices in receiving Internet streaming services. CAD improves on previous backlight scaling schemes by using the GPU instead of the CPU for online luminance adjustment and a backlight scaling algorithm with lower complexity compared to existing algorithms. We have implemented CAD on Android and experimented with more than 470 randomly selected YouTube video clips. The results show that CAD can effectively produce power savings while maintaining good streaming quality.

7. ACKNOWLEDGEMENT

We appreciate constructive comments from anonymous referees. The work is partially supported by High-Tech Research and Development Program of China ("863 - China Cloud" Major Program) under grant SQ2015AAJY1595, by China NSF under grant 61471217, by China Postdoctoral Science Fund under grant 2014M550735, and by NSF under grants CNS-0746649, CNS-1117300.

8. REFERENCES

[1] FFmpeg. `http://www.ffmpeg.org/`.
[2] Monsoon Power Monitor. `http://www.msoon.com/LabEquipment/PowerMonitor/`.
[3] OpenGL ES. `http://www.khronos.org/opengles/`.
[4] A. Carroll and G. Heiser. An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, 2010.

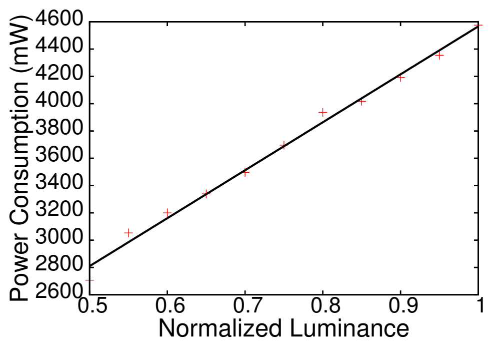

Figure 3: Display power consumption model.

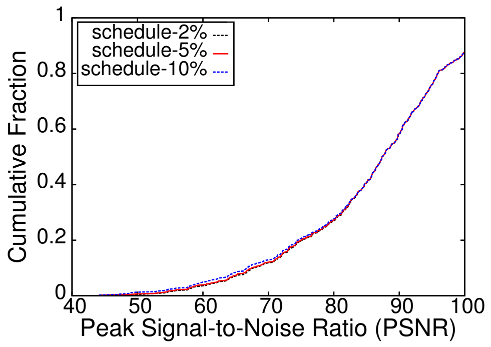

Figure 4: PSNR (dB) between the rendered video and the non-backlight-scaled video.

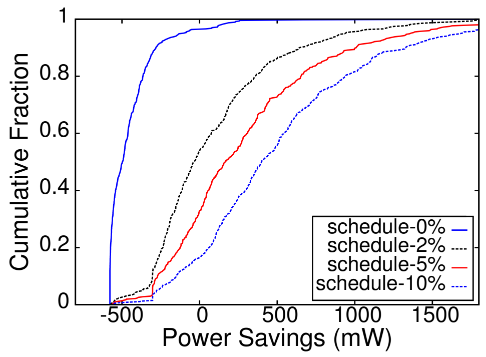

Figure 5: Estimated power savings (accounting for GPU power consumption) with backlight scaling computed under different settings.

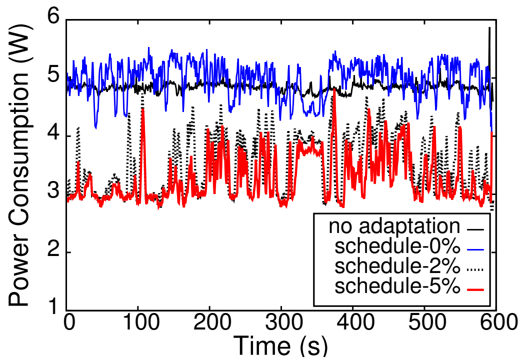

Figure 6: Measured results: power consumed by the entire device with backlight scaling.

[5] N. Chang, I. Choi, and H. Shim. Dls: dynamic backlight luminance scaling of liquid crystal display. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 12(8):837–846, Aug 2004.

[6] L. Cheng, S. Mohapatra, M. El Zarki, N. Dutt, and N. Venkatasubramanian. Quality-based backlight optimization for video playback on handheld devices. *Adv. MultiMedia*, 2007(1), Jan. 2007.

[7] W.-C. Cheng and M. Pedram. Power minimization in a backlit tft-lcd display by concurrent brightness and contrast scaling. *Consumer Electronics, IEEE Transactions on*, 50(1):25–32, 2004.

[8] H. Cho and O.-K. Kwon. A backlight dimming algorithm for low power and high image quality lcd applications. *Consumer Electronics, IEEE Transactions on*, 55(2):839–844, 2009.

[9] I. Choi, H. Shim, and N. Chang. Low-power color tft lcd display for hand-held embedded systems. In *Low Power Electronics and Design, 2002. ISLPED '02. Proceedings of the 2002 International Symposium on*, pages 112–117, 2002.

[10] P.-C. Hsiu, C.-H. Lin, and C.-K. Hsieh. Dynamic backlight scaling optimization for mobile streaming applications. In *Proceedings of the 17th IEEE/ACM international symposium on low-power electronics and design*, pages 309–314, 2011.

[11] C.-H. Lin, P.-C. Hsiu, and C.-K. Hsieh. Dynamic backlight scaling optimization: A cloud-based energy-saving service for mobile streaming applications. *Computers, IEEE Transactions on*, 63(2):335 – 348, Feb 2014.

[12] S. Pasricha, S. Mohapatra, M. Luthra, N. D. Dutt, and N. Venkatasubramanian. Reducing backlight power consumption for streaming video applications on mobile handheld devices. In *ESTImedia*, pages 11–17, 2003.

[13] M. Ruggiero, A. Bartolini, and L. Benini. Dbs4video: dynamic luminance backlight scaling based on multi-histogram frame characterization for video streaming application. In *Proceedings of the 8th ACM international conference on Embedded software*, pages 109–118. ACM, 2008.

[14] T. Simunic, L. Benini, P. Glynn, and G. De Micheli. Event-driven Power Management. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 20(7):840–857, 2001.

[15] P.-S. Tsai, C.-K. Liang, T.-H. Huang, and H. Chen. Image enhancement for backlight-scaled tft-lcd displays. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(4):574–583, 2009.

[16] J. Zhou, Y. Li, V. K. Adhikari, and Z.-L. Zhang. Counting youtube videos via random prefix sampling. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pages 371–380. ACM, 2011.