FingerShadow: An OLED Power Optimization based on Smartphone Touch Interactions

Xiang Chen[†‡], Kent W. Nixon[†‡], Hucheng Zhou[†], Yunxin Liu[†], Yiran Chen[‡]

Microsoft Research[†]
Beijing, China 100080
{huzho, yunliu}@microsoft.com

ECE Department, University of Pittsburgh[‡]
Pittsburgh, PA, USA 15261
{xic33, kwn2, yic52}@pitt.edu

Abstract

Despite that OLED screen has been increasingly adopted in smartphones to save power; screen is still one of the most energy-consuming modules in smartphones. Techniques such as local dimming are proposed to further reduce the power consumption of OLED screen, but it is hard to decide which part of the screen could be dimmed, and it often results in compromised user experience. Intuitively, when a user interacts with a smartphone via the touch screen, the screen areas are covered by the user's fingers and even some of the neighboring areas could be safely dimmed. Thus, in this paper, we propose FingerShadow, a new technique which does local dimming for the screen areas covered by user fingers to save more power, without compromising the user visual experience. We have studied 10 users' touch interaction behaviors and found that on average 11.14% of the screen were covered by fingers. For these 10 users, we estimate that FingerShadow can achieve 5.07%~22.32% power saving, averaging 12.96%, with negligible overhead. We discuss the challenges and future research work to implement FingerShadow in existing smartphone systems.

1. Introduction

Power is a paramount concern on smartphones and screen is one of the top power-hungry modules. To reduce power consumption of smartphone screen, Organic Light Emitting Diode (OLED) screen has been developed. Compared to the old generations of Super-twisted Nematic (STN) [1] screen and Liquid Crystal Display (LCD) [2] screen, OLED screen offers better display quality, lower power consumption, and the feasibility of manufacturing flexible and transparent screens [3][4]. Therefore, OLED is increasingly adopted in modern smartphones. However, even with low power OLED, screen is still one of the most power-hungry modules in smartphones, consuming near half the total energy in normal usage [5].

Different from STN and LCD, OLED's power consumption is color dependent. An OLED pixel is composed of three basic sub-pixels corresponding to RGB color space. Each sub-pixel is driven by an independent TFT driver circuit, and the emitting efficiency of the sub-pixels varies with different RGB colors. Taking advantage of this color-dependent nature, techniques such as local dimming and color remapping have been proposed to further reduce the power consumption of OLED screens. Local dimming proportionally lowers the RGB values, while color remapping changes power hungry colors to power friendly ones [2][6][7].

Arbitrary decision on where to apply local dimming or color remapping would compromise the user visual experience. Existing works utilized techniques, such as visual unfocused area analysis [2], driver panel division [6], and UI color remapping [7]. However, it is still hard to control the possible user experience interference, even with certain visual quality criteria such as SSIM [8] based on human visual perception or certain color compensations. Furthermore, they usually require considerable computation overhead in pre-analysis of display content and the frame buffer modification. These limitations make them hardly adopted.

We observe that when a user interacts with a smartphone via the touch screen, e.g., in using Twitter, Facebook, and many other interactive Apps, the screen is partially covered by the user's fingers. Intuitively, the areas covered by the fingers and even some of its neighboring areas could be safely dimmed or color-remapped, without compromising the user experience. Besides finger actions such as tap, swipe and scroll, users may also constantly hover their fingers over the screen for a long time, e.g., in scrolling a list of tweets. To this end, we propose FingerShadow, a new technique which applies local dimming to the screen areas covered by user fingers to save more power. As the user cannot see the screen areas covered by her or his fingers, FingerShadow is able to save power without compromising the user visual experience. FingerShadow does not need to analyze the content of the screen and thus imposes minimal computation overhead.

In this paper, we have studied 10 users' touch interaction behaviors in two popular Apps, Twitter and Facebook, and found that on average 11.14% of the screen

were covered by the fingers during the entire usage, with the maximum coverage area as high as 60% (Section 2). Using a power model, we estimate that FingerShadow can achieve about 5.07%~22.32%, averaging 12.96% screen power saving for these 10 users. This demonstrates the great performance of FingerShadow in saving screen power (Section 3). We further show that FingerShadow can be implemented with low overhead and discuss the challenges and future research work towards fully implementing FingerShadow in existing smartphone systems (Section 4).

2. Finger Operation Behavior Study

This section aims to study on how many optimization opportunities of FingerShadow can be applied in real Apps usage behaviors.

2.1 Methodology

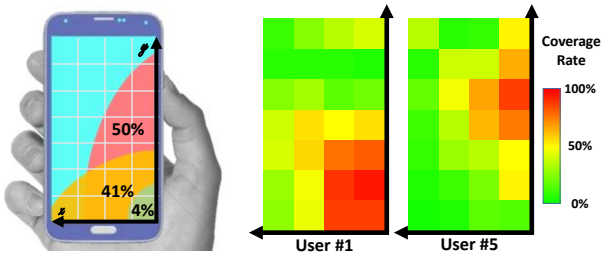
We study different finger behaviors from real App usage. To reduce bias, we select 10 different videos in YouTube from 10 different users that records real finger behaviors via camera, rather than usage behaviors from nearby friends. They record the usage in most popular Apps, Twitter and Facebook (for example, [9]). We cut these videos to only select a continuous data with the length of 2~3 minutes, and extract the frames with two frames per second (fps=2), .i.e., the gap between two continuous frames is half second. Larger fps will not help much that one finger action would span multiple frames. For each frame, we register the regions that are mainly covered by the finger.

The screen is divided into 4×7 regions to compute the finger coverage ratio of the entire screen. A region is treated as the basic area for analysis. Fig. 2(a) shows three typical movement patterns on the whole screen regions, including 1 vertical slide, 1 horizontal slide and 1 corner hover. In Fig. 2(a), the coverage ratios are about 50%, 41% and 4%, respectively.

Therefore, we can track all the finger movement patterns and time sessions to evaluate the finger coverage on the screen. We accumulated the coverage ratio of each screen region from all the frames. And the whole screen's coverage condition is shown in heat map. For a better understanding, we presented all the data as percentage ratio. Fig. 2(b) shows the finger usage heat maps for user #1, and #5, respectively. It is shown that different user could have different finger usage habits, including the hand holding position, preferred finger movement track and hover position.

2.2 Data Analysis

From the videos, 10 users overall finger operation behavior statistics are analyzed. The max coverage *time*,



(a) (b)

Fig. 1 Finger operation behavior methodology: (a) Screen division and finger movement coverage, (b) Total screen coverage heat maps from two users.

which indicates that how long the operation finger is touching or hovering on the screen with any coverage area, are 27%~92%, and the average coverage *time* is 59.72%. For the extreme case of 92%, which can be seen in user #1's heat map, the user prefers to rest his thumb on the right bottom corner. This indicates that, during the smartphone usage, there is a considerable time that people's fingers are on the screen, either touching or hovering. Meanwhile, the average *time* of a finger movement session is about 2.7s~7.1s.

When considering the screen coverage *area* when each finger movement (or gesture) is issued as shown in Fig 2(a), there is a significant screen coverage *area* about 27%~60% for each movement session. Averagely, each user would block 40.73% of the screen *area* for one finger gesture. The biggest data set comes from user #6, since he is operating the smartphone with right index finger and the screen is frequently covered by his partial palm.

Combining the finger movement *time* and *area*, from the whole operation video clip, the total coverage ratio for 10 users are is 4.65%~22.15% respectively, averaging 11.14%. It indicates that, averagely the user' finger is continuously covering about 11.14% of the screen areas when operating interactive Apps like Twitter or Facebook. For the proposed FingerShadow design, these coverage areas can be directly dimmed for power

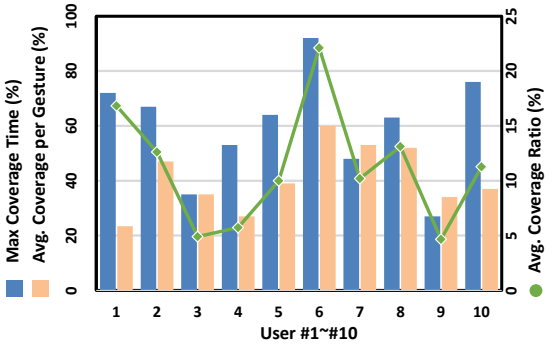


Fig. 2 Finger operation behavior statistic

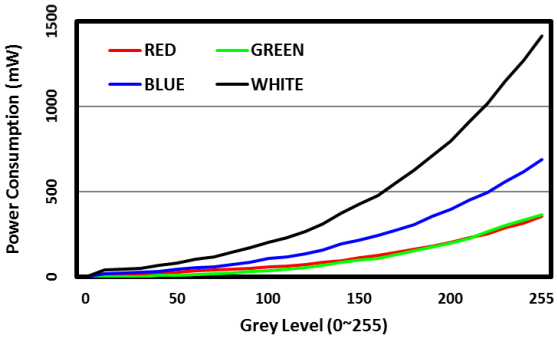


Fig. 3 The power model of Galaxy S5 OLED display panel

saving without user experience defection. Hence, comparing the 3 data sets from Fig. 2, we can tell that, each user has specific finger operation behavior. These behavior patterns could create significant operation opportunities for FingerShadow to optimize OLED power.

3. Performance Evaluation

Based on the behavior study, this section continues to evaluate the power savings of FingerShadow via both frame level simulation and in real-time usage track.

3.1 OLED Power Model

Before the evaluation, an OLED power model is necessary for data simulation and power consumption estimation. The OLED panel's power consumption is computed by the sum of each individual pixel's power consumption, which is the sum of its RGB sub-pixels. We have built a power model of Samsung Galaxy S5 (1920x1080) shown in Fig. 1, which is measured by Monsoon power monitor [10]. During the measurement, the screen brightness is set to be the maximum. The color value in x-axis scaled from 0 to 255 presents the whole brightness range the device can generate. The power value in y-axis indicates that the whole display panel's power consumption, when display a specific monochromatic color. Accordingly, the power consumption of each pixel with a given RGB value can be computed by the entire screen power corresponding to that RGB value divides the number of pixels (1920x1080). From the power model shown in Fig. 3, we can tell that, although the display panel is very power efficient in the low brightness range, the power consumption increases exponentially and makes the display screen the biggest power contributor (considering the whole devices' power consumption is ~ 2W).

(a) (b) (c)

Fig. 4 Different local dimming policy examples

3.2 FingerShadow Polices Evaluation

In FingerShadow, when the finger is detected either touching or hovering above the screen, which will be discussed in Section 4.1. It will apply local dimming to the covered screen areas and even some of the neighboring areas. Different local dimming policies, with different dimmed areas, achieve different savings on the same frame. We thus evaluate the three local dimming policies from conservative to aggressive.

In Fig. 4, the frame's background is a screenshot from the Facebook App, and its screen power is about 1,047mW based on Galaxy S5's power model. When the finger is touching the screen, the finger's length, width and direction can be quickly calculated by the touch area [11]. They are used to compute the finger coverage area. Once the scree area for local dimming is decided, FingerShadow directly shutdowns the color emitting pixels in that area.

There are three local dimming policies evaluated:

(1) Only the area exactly covered by finger is dimmed (rigid dimming), and the corresponding RGB value is changed to 0 with complete black color. Note that the touch area is generally smaller than the actual finger size. For instance in Fig. 4(a), the rigid coverage area ratio is about 6.1% of the screen, and about 9.3% of screen power saving can be achieved.

(2) The neighboring areas could also be safely dimmed other than the area exactly under the finger (neighbor dimming), as shown in Fig. 4(b), since finger operation would distract user's attention around the finger area [12]. The neighboring area is dimmed with half of the original RGB value. By enlarging the area via neighbor dimming, the power saving is increased to 14%.

(3) A much more aggressive dimming policy is evaluated. For finger movement like swipe and scroll, the user would ignore the area on the finger gesture path [12] (track dimming). The finger movement specs is easy to compute, such as swipe radius, scroll length can be well predicted after short time of usage [13]. For instance, the track of finger gestures, shown in Fig. 4(c), is dimmed. A radius area is selected for the moving finger. And for a single frame, the power saving ratio via track dimming is further increased into 26%.

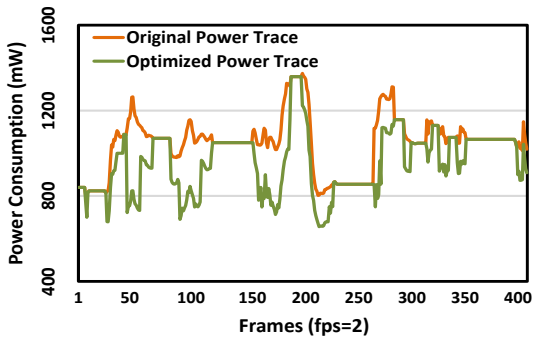


Fig. 5 Finger operation behavior statistic

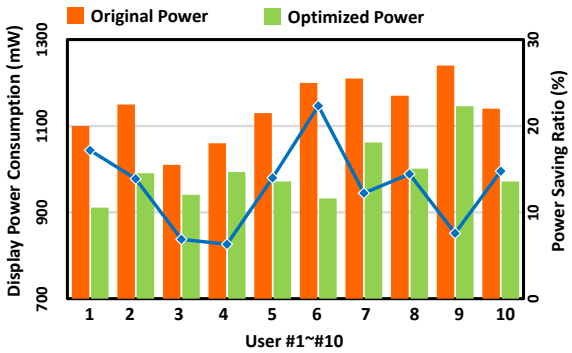


Fig.6 FingerShadow performance evaluation

3.3 Real-time Performance Evaluation

Based on the above local dimming policy analysis, we continue to evaluate FingerShadow in practical real-time scenario. However, due to the security consideration, Android operating system limits the finger movement tracking or modification for user Apps, except system UI activities. As a result, FingerShadow currently cannot directly apply to Apps like Facebook or Twitter. The possible solution is discussed in Section 4.4, and in this section, we built a simple test benches to mimic the UI of Facebook, to simulate the real Facebook usage from 10 users described in Section 2.1. When a user's finger movement trace is inputted, the test bench replays the finger gesture, such as tap and slide. Then the test bench could generate corresponding display frames with a Facebook screenshots pool. Afterwards, FingerShadow applies the track dimming policy to evaluate the real power savings.

For example, the evaluation from user is shown in Fig. 5. There are 400 frames are extracted from video with 200 seconds (fps=2). The original power consumption curve is shown as yellow color. It can be seen that the display power varies significantly if the display content changes. The optimized power consumption curve is shown as green color. It shows obvious power reduction, and different finger gestures with different coverage area ratios have different power savings, their different coverage ratio. For some traced frames, the original power consumption and the optimized power consumption stay the same, which indicates that there is neither finger touch nor the finger hover on the screen. In this example, the original average power consumption is 1,048.84mW, and the average after optimization is reduced to 952.48mW. FingerShadow can achieve about 10.52% of the power savings.

Fig. 6 depicts the power savings of FingerShadow for all the 10 users, that we apply the same methodology described above. Due to different user activities, the original screen power consumption is varying between 1,010mW~1,240mW, shown as red bar in Fig. 6. The optimized power consumption is shown in green bar, varying from 910mW~1,145mW, with about 5.07%~ 22.32% power saving ratio, and 12.96% in average. The biggest data comes from user 6, which complies with the opportunity study shown in Fig. 2 that user 6 has the biggest screen coverage ratio.

Since FingerShadow's power saving is directly user behavior dependent, the power optimization result almost complies with the finger behavior study shown in Section 2. However, due to complex display content and practical local dimming traces, the power saving ratio doesn't equals the coverage ratio.

4. Discussion

In this work, the behavior study and power evaluation is still remained by the level of data collection and power model based simulation. We are still working on the FingerShadow integration on smartphone. This discusses the findings and challenges in implementation.

4.1 Finger Position Detection

Despite of default touch screen sensor, which can call back touch event and provide fingertip area, we also need to detect finger's "hover" status. In present smartphones, like Galaxy and Xperia series, capacity touch screen are improved with another layer of hover sensors, which generates stronger sensing signals and allows accurate detection of the hovering finger further away. On Galaxy S5, by reconfiguring the screen sensor sensitivity in Android, we found that the screen can generally sense the hovering finger over 1.5cm above the screen [14]. By analyzing the touch/hover area and shape, the finger vector can be built; therefore the finger coverage area can be well detected [11].

During touch operation considerable power overhead may be introduced by the system, for related UI and App response computation. However, measured by the Monsoon power monitor, the power consumption overhead of the hover senor is < 40mW.

4.2 Adaptive Local Dimming Policies

There could be more local dimming policies, besides the three policies we evaluated, which are based on previous researches. It would be interesting to learn the relationship between the finger position and the user visional focus area on the smartphone screen, not only based on certain heuristics. Besides, eye tracking devices in the future could also be helpful to accurately compute the dimmed areas. And the dimming degree on RGB value modification could also be configurable or dynamic adjusted, which strikes a better tradeoff between power saving and user experience feedback. The color remapping is also interesting, but how to efficiently decide the right color remains future work.

Other than used passively, FingerShadow can also be used proactively. Essentially, FingerShadow provides such flexibility. For instance, the user can proactively cover a large part of the screen for aggressive power saving without any help of other tools, in the scenarios that the smartphone is going to be out of power while he/she still wishes to use it even with certain experience compromise.

4.3 Computation Overhead

The major computation involved in the FingerShadow is the display frame buffer rending. However, compare to previous research, FingerShadow has several advantages. First, the dimming area is located by the finger touch/hover operation, and the sensor call back process's calculation load is ignorable. Second, the pixel rending process doesn't require intensive computation load, the target area could be directly turned down or dimmed to the predefined level. Third, the dimming layer introduced by FingerShadow is handled by dedicated display controller rather than GPU, which is much cheaper with only about 11.14% of the workloads. These overheads are totally ignorable.

Such dedicated display controllers are already integrated in recent smartphones with OLED screen, such as Samsung Galaxy S4 and S5. Dimming algorithms are mature enough to be implemented in image processing by an integrated module called MDNIe (Mobile Digital Natural Image engine) [15], which can dynamically adjust the brightness, sharpness, contrast and remap color tone of the OLED screen at pixel-level. The corresponding APIs have also been released to the software developers, allowing for customization. We will port the FingerShadow on top of it in the future.

4.4 Framework Integration

In this work, the FingerShadow evaluation is based on only two apps, Facebook and Twitter, and a self-made Facebook test bench. To mitigate the limitation that the finger movement tracking is not exposed for user Apps in Android operating system, we consider adding FingerShadow as a new component in Android App framework, that developer can integrate FingerShadow for their Apps. Another workaround approach is to use the eye-track techniques instead of finger track.

5. Conclusion

In this paper, we proposed a new screen power saving technique based on user touch interactions, FingerShadow, which applies local dimming for the screen areas covered by user fingers without compromising user experience too much. It shows promising potential with about 12.96% in average of screen power saving, with 22.32% at most. In the future, more general useful principles from user interactions would be applied for display power saving. For instance, when user swipes the phone, the screen could be turned off; or the screen could be completely dimmed or even turned off for driving App usage, it is only turned on or switched to normal when the voice prompts is on.

References

[1] T. Scheffer, *et al*. "Supertwisted Nematic (STN) Liquid Crystal Displays," Materials Science, 1997.

[2] A. Iranli, et al. "HVS-Aware Dynamic Backlight Scaling in TFT LCD's." IEEE Trans. on *VLSI Sys.*, 2006.

[3] W. Graupner, *et al*. "High-resolution Color Organic Light-emitting Diode Micro-display Fabrication Method," *Int'l Society for Optical Engineering*, 2000.

[4] G. Gustafsson, *et al*. "Flexible Light-emitting Diodes Made from Soluble Conducting Polymers," *Nature*, 1992.

[5] X. Chen, *et al*. "Fine-grained Dynamic Voltage Scaling on OLED Display," *Asia and South Pacific Des. Auto. Conf.*, 2012.

[6] X. Chen, *et al*. "Quality-retaining OLED Dynamic Voltage Scaling for Video Streaming Applications on Mobile Devices," *Des. Auto. Conf*., 2012.

[7] M. Dong, *et al*. "Power-saving Color Trans-formation of Mobile Graphical User Interfaces on OLED-based Displays," *Int'l Symp. on Low Power Elec. and Des.*, 2009.

[8] Z. Wang, *et al.* "Image Quality Assessment: From Error Visibility to Structural Similarity," IEEE Trans. on *Image Proc.*, 2004.

[9] http://www.youtube.com/watch?v=MykiGpRlu2Y

[10] Monsoon Solutions Inc. Power Monitor. http://www.msoon.com/LabEquipment/PowerMonitor/

[11] F. Wang, *et al*. "Empirical Evaluation for Finger Input Properties in Multi-Touch Interaction," SIGCHI Conf. on *Human Factors in Computing Sys*, 2009.

[12] M. Negulescu, *et al*. "Tap, Swipe, or Move: Attentional Demands for Distracted Smartphone Input," *Int'l Working Conf. on Advanced Visual Interfaces*, 2012.

[13] M. Franck, *et al*. "Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication," *Infor. Forensics and Security*, 2013.

[14] Tuning the Touchscreen Sensitivity, http://forum.xda-developers.com/showthread.php?t=844216

[15] MDNIe: Digital Natural Image Engine, www.samsug.com.