Apple Inc. v. Samsung Electronics Co. Ltd. et al Doc. 434 Att. 2

EXHIBIT 2

Dockets.Justia.com

Power Modeling of Graphical User Interfaces on OLED Displays

Mian Dong Yung-Seok Kevin Choi Lin Zhong
Department of Electrical & Computer Engineering, Rice University, Houston, TX 77025
{dongmian, ykc1,lzhong}@rice.edu

ABSTRACT
Emerging organic light-emitting diode (OLED)-based displays obviate external lighting; and consume drastically different power when displaying different colors, due to their emissive nature. This creates a pressing need for OLED display power models for system energy management, optimization as well as energy-efficient GUI design, given the display content or even the graphical user interface (GUI) code. In this work, we present a comprehensive treatment of power modeling of OLED displays, providing models that estimate power consumption based on pixel, image, and code, respectively. These models feature various tradeoffs between computation efficiency and accuracy so that they can be employed in different layers of a mobile system. We validate the proposed models using a commercial QVGA OLED module. For example, our statistical learning-based image-level model reduces computation by 1600 times while keeping the error below 10%, compared to the more accurate pixel-level model.

Categories and Subject Descriptors
I.6.5 [**Simulation and Modeling**]: Model Development

General Terms
Algorithms, Measurement, Human Factors

Keywords
OLED Display, Graphic User Interface, Low Power

1. INTRODUCTION
Energy consumption is an important design concern for mobile embedded systems that are battery-powered and thermally constrained. Displays have been known as one of the major power consumers in mobile systems [1-4]. Conventional liquid crystal display (LCD) systems provide very little flexibility for power saving because the LCD panel consumes almost constant power regardless of the display content while the external lighting dominates the system power consumption. In contrast, the power consumption by emerging organic light-emitting diode (OLED)-based displays [5] is highly dependent on the display content because their pixels are emissive. For example, our measurement shows that a commercial QVGA OLED display consumes 3 and 0.7 Watts showing black text on a white background and white text on a black background, respectively. Such dependence on display content leads to new challenges to the modeling and optimization of display power consumption. First, it makes it much more difficult to account display energy consumption in the operating system for optimized decisions. Second, GUI designers will have a huge influence on the energy cost of applications, which is not their conventional concern. As a result, there is a great need of OLED display power models for use at different layers of a computing system and different stages of system design.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DAC'09, July 26-31, 2009, San Francisco, California, USA.
Copyright 2009 ACM 978-1-60558-497-3/09/07….5.00.

In this work, we provide a comprehensive treatment of OLED display power modeling. In particular, we make three contributions by addressing the following research questions.

First, *given the complete bitmap of the display content, how to estimate its power consumption on an OLED display with the best accuracy?* An accurate *pixel-level model* is the foundation for modeling OLED display power. We base our pixel-level model on thorough measurements of a commercial QVGA OLED module. In contrast to the linear model assumed by previous work [6], we show that the power consumption is nonlinear to the intensity levels of the color components. Our nonlinear power model achieves 99% average accuracy against measurement of the commercial OLED display module. This is presented in Section 4.

Second, *given the complete bitmap of the display content, how to estimate the power consumption with as few pixels as possible?* This *image-level model* is important because accessing information of a large number of pixels can be costly due to memory and processing activities. We formulate the tradeoff as a sampling problem and provide a statistical optimal solution that outperforms both random and periodical sampling methods. Our solution achieves 90% accuracy with 1600 times reduction in sampling numbers. This is presented in Section 5.

Third, *given the code specification of a GUI, how to estimate its power consumption on an OLED display?* This *code-level model* is important to GUI designers as well as application and system based energy management. We use the code specification to count pixels of various colors and calculate the power consumption of the OLED display. Our model guarantees over 95% accuracy for 10 benchmark GUIs. This is presented in Section 6.

To the best of our knowledge, this is the first public study that addresses the three research questions above. The modeling methods presented here provide powerful mechanisms for operating systems and applications to construct energy-conserving policies for OLED displays. They will also enable GUI designers of mobile systems to build adaptable and energy-efficient GUIs; and empower end users to make informed tradeoffs between battery lifetime and usability.

The rest of the paper is organized as follows. We provide background and address related work in Section 2. We describe the experimental setup used in this study in Section 3. From Sections 4 to 6, we present the power models and their experimental validations. We conclude in Section 7.

2. BACKGROUND AND RELATED WORK
How Display Works. Figure 1 illustrates the relationships between the display and the rest of the system. The *main processor*, or *application processor*, runs the operating system (OS) and

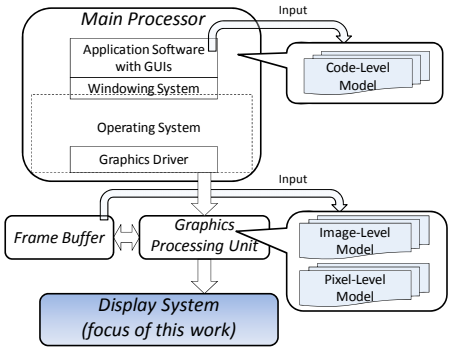

Figure 1. Display system in a typical mobile system. The three power models proposed require input of different abstractions and provide different accuracy-efficiency tradeoffs

application software with GUIs. Note the windowing system can be either part of the OS, e.g. Windows, or a standalone process, e.g. X Window under Linux. the *graphics processing unit*, often including a graphics accelerator and a LCD controller in a system-on-a-chip for mobile devices, generates the bitmap of the display content and stores it in a memory called *frame buffer*; the bitmap is sent to the display for displaying. Each unit of this bitmap is described using sRGB, or standard RGB, color space, in which a color is specified by (R, G, B), the intensity level of red, green and blue component. Because sRGB uses gamma correction to cope with the nonlinearity introduced by cathode ray tube (CRT) displays, the intensity level of each component and its corresponding luminance follow a nonlinear relation [7].

OLED Display. Organic light-emitting diode or OLED [5, 8] is an emerging display technology that provides much wider view angle and higher image quality than conventional LCDs . The key difference in power characteristics between an OLED display and a LCD is that an OLED display does not require external lighting because its pixels are emissive. Each pixel of an OLED display consists of three types of devices, corresponding to red, green and blue components, respectively. Moreover, the red, green, and blue components of a pixel have different luminance efficacies. As a result, the color of a pixel directly impacts its power consumption and GUI has a significant impact on the display power. In contrast, color only has negligible power impact on LCDs and illumination of external lighting dominates. OLED displays and LCDs have a very similar organization, including a panel of addressable pixels, LCD or OLED, control circuitry that generates the control and data signals for the panel based on display content, and interface to the graphics processing unit. In this work, we address the power consumption of the display and focus on the variance introduced by the OLED panel. Our power models take input from different places of the system and can be implemented either as a software tool, an operating system module, or an extra circuit.

We focus on the power consumption by a constant screen because of the following two reasons. First, a display spends most time displaying a constant screen, even for high-definition video that requires 30 updates per second. Second, our measurement showed that the power consumption by an OLED display during updating is close to the average of those by the constant screens before and after the updating. Therefore, the energy contribution by OLED display updating is very small and can be readily estimated from the power models of a constant screen. However, we note that

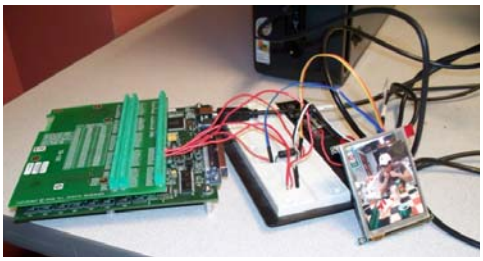

Figure 2. Measurement setup of the QVGA OLED module used in our experimental validation

screen updating may incur considerable energy overhead in graphics processing unit, frame buffer, and data buses, which are out of the scope of this work.

Related Work. HP Labs pioneered energy reduction for OLED displays [6, 9, 10]. Yet no real OLED displays were reported in the work. The power model employed was pixel-level, thus expensive to use, and incorrectly assumed a linear relationship between intensity levels of color components and power consumption. As we will show, the relationship is indeed nonlinear. The IBM Linux Wrist Watch was one of the earliest users of OLED displays [11]. The work, however, did not employ or provide a power model for the OLED display. There is also a large body of work on energy optimization of conventional LCD systems [2, 4, 12-23]. While many of the proposed techniques may be applied to OLED displays, they are orthogonal to the power modeling techniques presented in this work.

3. EXPERIMENTAL SETUP

Benchmark GUI Images. To evaluate the proposed power models, we collect 300 GUI images from three Windows Mobile-based cell phones, HTC Touch, HTC Mogul, and HTC Wizard, all with a resolution of QVGA (240×320). On each phone, we exhaust all the varying GUI screens, representative of everyday use of a smart phone (e-mail, web browser, games, etc). We also capture screens with different color themes available.

Measurement Setup. For our experimental validation, we use a 2.8" OLED QVGA display module with an integrated driver circuit, µOLED-32028-PMD3T, from 4D Systems [24]. We connect it to a PC using a micro USB interface, which also supplies power to it. Through the USB, we can send commands to the OLED module to display images. The display module employs a standard 16-bit (5,6,5) RGB setting. That is, there are five, six, and five bits to represent the intensity of red, green, and blue component, respectively. In this work, we call their numbers the *intensity level* or *value* of the three color components.

We obtain the power consumption of the OLED module by measuring the current it draws from the USB interface and its input voltage. Figure 2 shows the measurement setup with a DAQ board from Measurement Computing and the OLED module. To overcome the variance among different measurements of the same image, we take an average of 1000 measurements for each image.

4. PIXEL-LEVEL POWER MODEL

We first present a pixel-level power model that estimates the power consumption of OLED modules based on the RGB specification of each pixel. It is intended to be the most accurate and constitutes the baseline for models based on more abstract descriptions of the display content.

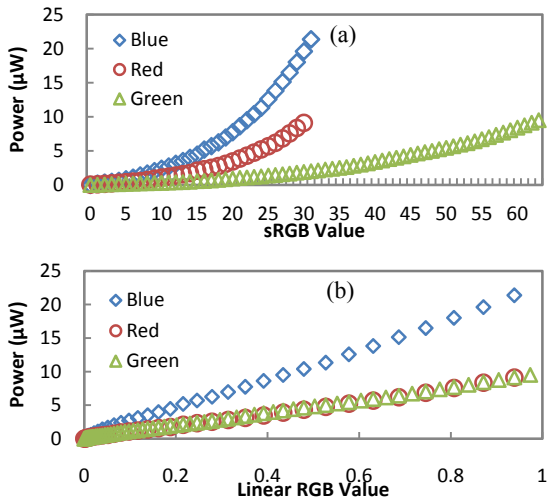

Figure 3. Intensity Level vs. Power Consumption for the R, G, and B components of an OLED pixel

Power Model of OLED Module. We model the power contributed by a single pixel, specified in (R, G, B), as

$$P_{pixel}(R, G, B) = f(R) + h(G) + k(B),$$

where $f(R)$, $h(G)$ and $k(B)$ are power consumption of red, green and blue devices of the pixel, respectively. And the power consumption of an OLED display with n pixels is

$$P = C + \sum_{i=1}^{n}\{f(R_i) + h(G_i) + k(B_i)\}.$$

Note that the model includes a constant, C, to account for static power contribution made by non-pixel part of the display, which is independent with the pixel values. This pixel model has a generic form that applies to all the colorful OLED display modules.

Power Models for RGB Components. We obtain C by measuring the power consumption of a completely black screen. To obtain $f(R)$, we fill the screen with colors in which the green and blue components are kept zero and the red component, R, varies from 0 to 31, enumerating every possible intensity level. For each measurement, we subtract out C to get just the power contribution by the red pixel component, or $f(R)$. We obtain $h(G)$ and $k(B)$ similarly. Figure 3 (a) presents the measured data for all three components. Apparently, the power contribution by pixel components is a nonlinear function, instead of a linear function as assumed in [6], of the intensity level. The nonlinearity is due to the gamma correction in sRGB standard. After transforming the intensity level into linear RGB format, which is the indication of luminance, we obtain a linear relation between pixel power consumption and intensity, as shown in Figure 3(b).

The measured data can be directly used to estimate the power consumption of displaying an image on the OLED display through simple table lookup. One can also apply curve fitting to obtain close-form functions for $f(R)$, $h(G)$, and $k(B)$.

Estimation vs. Measurement. Figure 4 shows the histogram of the error of the estimation against the measurement for the 300 benchmark images. It shows that 63% of the samples have no more than 1% error and 93% have no more than 3% errors. The average absolute error is only 1%.

It is important to note that although we derived the pixel-level power model from a specific OLED display, the methodology can be largely extended to other OLED displays with a similar RGB organization.

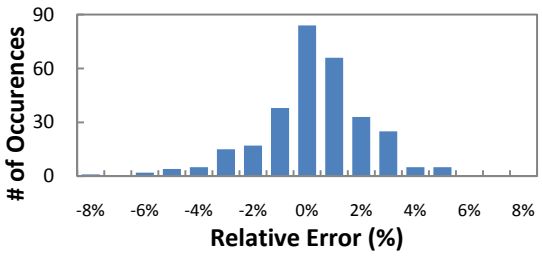

Figure 4. Histogram of percent errors observed for the 300 benchmark GUIs using our pixel-level power model

5. IMAGE-LEVEL POWER MODEL

We next present models that estimate power consumption given the image to display. They are important for a system to assess the display power cost when the pixel information is known, e.g. from the frame buffer. A straightforward image-level model can be a simple application of the pixel-level model to all pixels. Such a method, unfortunately, is exceedingly expensive. Modern displays can have hundreds of thousands or millions of pixels. The cost of a large number of pixel power calculations and the overhead of accessing the frame buffer can be prohibitively high for the system. Our solution to this problem is to estimate the power based on a small subset of pixels, or *sampling*.

5.1 Problem Formulation

The power consumption by an image on an OLED display can be described by a vector of N elements, i.e., $y = (y_1, y_2, \dots, y_N)^T$, in which y_n denote the power consumption of the i-th pixel, $i = 1,2, \dots, N$. The total display power can be calculated as $Power = \mathbf{1}^T y = \sum_{i=1}^{N} y_i$.

Sampling is to select K pixels out of N and use the average of K samples to approximate the average of all N pixels. For each pixel, we use a random variable to indicate whether this pixel is sampled or not, i.e., $X_i = 1$ when the i-th pixel is sampled; otherwise $X_i = 0$. Thus, the sampling of an image can be represented by a vector $X = (X_1, X_2, \dots, X_N)^T$. And denote the joint probabilistic distribution of them as $f_X(x) = f_{X_1,X_2,\dots,X_N}(x_1, x_2, \dots, x_N)$. Given an image, y, the estimation error can be calculated as

$$\varepsilon = E_X\left[\left(\frac{X^T y}{K} - \frac{\mathbf{1}^T y}{N}\right)^2\right] = E_{X_1,X_2,\dots,X_N}\left[\left(\frac{\sum_{i=1}^{i=N} X_i y_i}{K} - \frac{\sum_{i=1}^{i=N} y_i}{N}\right)^2\right]$$

$$= \int \left(\frac{\sum_{i=1}^{i=N} x_i y_i}{K} - \frac{\sum_{i=1}^{i=N} y_i}{N}\right)^2 f_{X_1,X_2,\dots,X_N}(x_1, x_2, \dots, x_N)dx.$$

Therefore, an optimal sampling given the image can be obtained by finding $f_X(x)$ that leads to the minimal ε, or $f_X^*(x) = \arg\min_{\forall f_X(x)} \varepsilon$.

Now, we consider all possible images by treating the power consumption of an image as a random vector, $Y = (Y_1, Y_2, \dots, Y_N)^T$, and denote the joint probabilistic distribution as $g_Y(y) = g_{Y_1,Y_2,\dots,Y_N}(y_1, y_2, \dots, y_N)$. Then the optimal sampling for all images can be found as $f_X^*(x) = \arg\min_{\forall f_X(x)} E_Y[\varepsilon]$, i.e., finding a probability simplex $p = (p_1, p_2, \dots, p_{\binom{N}{K}})^T$ such that $E_Y[\varepsilon]$ is minimal, where $p_j = prob(X = x^{(j)})$ and $x^{(j)}$ is the j-th possible combination of X. This is a linear programming problem, i.e.,

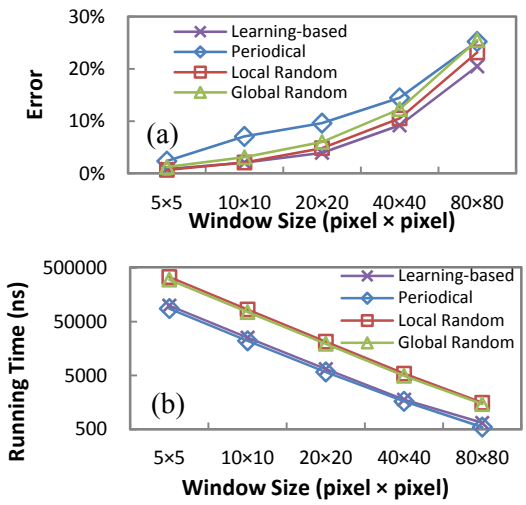

Figure 5. Comparison of four different sampling methods

$$\min \int \sum_{j=1}^{\langle N \rangle_K} p_j \left(\frac{\sum_{i=1}^{i=N} x_i^{(j)} y_i}{K} - \frac{\sum_{i=1}^{i=N} y_i}{N} \right)^2 g_{Y_1,Y_2,\ldots,Y_N}(y_1, y_2, \ldots, y_N) dy$$

$$\text{s.t.} \quad \sum_j p_j = 1 \text{ and } p_j \geq 0$$

It is, however, impractical to obtain the joint distribution $g_{Y_1,Y_2,\ldots,Y_N}(y_1, y_2, \ldots, y_N)$, because the dimension is too high. Therefore, we transform the objective function to eliminate the join distribution as below.

$$\int \sum_{j=1}^{\langle N \rangle_K} p_j \left(\frac{\sum_{i=1}^{i=N} x_i^{(j)} y_i}{K} - \frac{\sum_{i=1}^{i=N} y_i}{N} \right)^2 g_{Y_1,Y_2,\ldots,Y_N}(y_1, y_2, \ldots, y_N) dy$$

$$= E_X \left[\left(\frac{X}{K} - \frac{\mathbf{1}}{N} \right)^T E_Y[YY^T] \left(\frac{X}{K} - \frac{\mathbf{1}}{N} \right) \right] = \sum_{j=1}^{\langle N \rangle_K} p_j b^{(j)T} \bar{A} b^{(j)}$$

in which $\bar{A} = E_Y[YY^T]$ and $b_i^{(j)} = \frac{x_i^{(j)}}{K} - \frac{1}{N}$, for $i = 1,2, \ldots, N$.

Therefore, the key to an accurate estimation is to obtain a good knowledge of the expectation of outer products of all possible images, which can be *statistically learned* from a set of training images by using the mean of the outer products of the training images to approximate the expectation $\bar{A}$.

5.2 Practical Learning-based Sampling

There are two critical barriers for the practical implementation of the statistically optimal sampling. First, the linear programming problem described above is very difficult, if possible, to solve because of the extremely high dimension, i.e. $\langle N \rangle_K$ or *N*-choose-*K*. To address this issue, we divide the image into much smaller windows, and then solve the linear programming problem for each window. By preparing a training set for the window at every position within an image, we are able to obtain an optimal sampling solution for every window. Second, the solution requires generating a large number of random numbers, which is expensive for mobile embedded platforms. Therefore, instead of random sampling, we decide to use deterministic sampling method in which we seek to solve a combinatorial problem to find the samples for each window such that $b^{(j)T} \bar{A} b^{(j)}$ achieves the minimal. That is,

$$\min \left(\frac{X}{K} - \frac{\mathbf{1}}{N} \right)^T \bar{A} \left(\frac{X}{K} - \frac{\mathbf{1}}{N} \right)$$

$$\text{s.t. } x_i = 1 \text{ or } 0, \text{ and } \sum_i x_i = K$$

Another practical issue is the number of samples to take from each window, or *K* in the formulation above. Given a fixed sampling rate, or the total number of samples, there are two strategies. One can increase the window size, therefore reduce the number of windows, but allow more samples per window. Or one can reduce the window size, therefore increase the number of windows, but allow fewer samples per window. The second strategy obviously is more efficient because the computational load increases much faster with the window size than with the number of samples per window. We also experimentally compared the accuracy of these two strategies. We found that there is no difference in accuracy between them. Therefore, we take the second strategy in our image-level mode, i.e. a single sample is taken from each window.

5.3 Experimental Results

To evaluate our proposed learning-based sampling method (Learning-based), we compare its performance against three other sampling methods, described below.

- Periodical sampling: an image is divided into non-overlapping windows of the same size, exactly the same as Learning-based sampling but the bottom right pixel of each window is selected.
- Local Random sampling: one pixel is randomly selected from each window.
- Global Random sampling: pixels are randomly selected from the whole image. The number of pixels is kept the same as the number of windows in the first three methods.

Collectively these benchmark sampling methods will highlight the effectiveness of the design decisions made in Section 5.2. It is important to note that for learning-based sampling, we employ a bootstrapping method to improve the reliability of accuracy evaluation because training is involved. That is, we divide the 300 benchmark images evenly into 10 groups, i.e. 30 in each. Then we use nine groups as training set and test the 10th group. We repeat the process 10 times using different groups as the test group and the report the average accuracy.

We investigate the tradeoff between sampling rate and accuracy by varying the window size from 5×5, 10×10, 20×20, 40×40, to 80×80. This leads to a sampling rate between 1/25 and 1/6400.

Figure 5 (a) presents the tradeoffs between estimation error and window size over the 300 benchmark GUIs described in Section 3. It clearly shows that as window sizes increases, or sample rates decreases, the estimation error increases, for all four sampling methods. Figure 5 (a) clearly shows that our learning-based method achieves the best tradeoffs between accuracy and sampling rate. When window size is 40×40 and a 1600 times reduction in pixels needed for power estimation, learning based sampling method achieves accuracy of 90%.

Figure 5 (b) presents the run-time of all four sampling methods on a Lenovo T61 laptop with a Core 2 Duo T7300 2GHz processor and a 2GB memory. It clearly demonstrates the advantage in efficiency of deterministic methods, i.e. Learning-based and Periodical. With the comparable accuracy, Learning-based sampling is at least three times faster than the random methods, both Global and Local. This will lead to significant efficiency improvement when the power model is employed in mobile embedded systems for energy management and optimization.

In summary, our learning-based sampling method achieves the best tradeoff between accuracy and efficiency. We note that the learning-based method achieves this through training, which can

```
for i = 2 to n
    for j = 1 to i − 1
        if O_i ∩ O_j ≠ Ø
            update(pl_j);
        end
    end
end
for k = 1 to n
    pl_GUI = merge(pl_GUI, pl_k);
end
```

Figure 6. Power estimation for a composition of GUI objects

be compute-intensive. However, training can be carried off-line and therefore does not consume any resource at run-time.

6. Code-Level Power Model

Both pixel and image-level models require that the RGB information is available for all pixels of the display content. In this section, we present a power model that is based on the code specification of the display content. This is possible because graphical user interfaces (GUIs) are usually described in high-level programming languages, e.g. C# and Java, and are highly structured and modular. The code-level model can be even more efficient than the sampling-based image-level power model because they do not need to access the frame buffer. Therefore, it can be readily employed directly by the application or the operating system. In addition, they also provide a tool for GUI designers to evaluate their designs at an early stage.

We take an object-oriented approach because modern GUIs are composed of multiple objects, each with specified properties, e.g. size, location, and color. Moreover, GUI programming is also object-oriented. Developers rarely specify the graphics details; instead they extensively reuse a "library" of customizable common objects, e.g. buttons, menus, and lists. They customize these objects by specifying their properties and their relationship with each other within a GUI. Our methodology for code-based power modeling is first estimating power contribution by individual GUI objects based on their properties and then estimating the total power based on the composition of these objects.

6.1 Power by GUI Object

We can view a GUI object as a group of pixels with designated colors. By accounting the number of pixels with each color that appear in an object, we can obtain its power consumption using the pixel-level power model. To obtain the estimated power for the object, we enhance the object class with a pixel list property that records the (R, G, B) values and pixel number N of each color. Therefore, by utilizing our pixel-level power model, we can calculate the power consumption of an object with K colors as

$$P_{object} = \sum_{i=1}^{K} N_i \times P_{pixel}(color_i).$$

In most cases, a GUI object only has three colors, i.e., border color, background color and foreground (text) color. All three can be obtained from the GUI object's properties. Knowing the geometric specification of the object, usually rectangular, we can calculate the number of pixels for both background and border colors. Then we create a pixel list of two entries for this object, which includes the (R, G, B) values of colors and their corresponding pixel numbers. By extending the pixel list, we are able to handle more complicated objects with arbitrary shapes and more colors.

Text is a special object property and needs a special treatment. Unlike the color information available from the object properties, the pixel number of text is not easy to obtain. Thus, we build a library for all the ASCII characters; the library contains the number of pixels of each character based on its font type and size. Thus, we are able to account the total number of pixels of a whole text string, which we should subtract from the background pixel number. For our experiments, we have constructed the library for Times New Roman and Tahoma of font sizes 12 and 9, which are the most common on Windows Mobile devices.

6.2 Power by Composition of Objects

A GUI usually consists of multiple objects. Simply aggregating the power of all of the objects is not accurate enough because they may overlap with each other. To consider the effect of overlapping, we maintain a pixel list ($\boldsymbol{pl_{GUI}}$) for the whole GUI based on the pixel list of each objects. We first sort the objects from back to front in the GUI. Denote the sorted object list includes $\boldsymbol{n}$ objects $\boldsymbol{O_1, O_2, \ldots, O_n}$, with the sequence from the back to the front, and the pixel list of $\boldsymbol{O_i}$ is $\boldsymbol{pl_i}$. As shown in Figure 6, we go through all the objects one by one while checking their overlapping situation. If two objects overlap with each other, we update the pixel list of the object in the back by subtracting the number of pixels being covered. Finally, we merge all the pixel lists by combining the pixels with the same color together. We describe how we deal with two special effects of GUIs below.

Dynamic Objects. Some GUI objects can have multiple states. For example, a radio button has two states, checked and non-checked; a menu has even more states when different items are activated. For these objects, we maintain a pixel list for each state and treat each state as an individual object in the procedure described above.

Themes. Many operating systems support color themes that contain pre-defined graphical details, such as colors and fonts, so that GUIs of different applications will follow a coherent style. For instance, in Windows Mobile and C#, BackColor and ForeColor can be either specified using (R, G, B) or selected from a set of pre-defined colors, such as Window and ControlText. These system colors are determined by the theme of Windows Mobile. Fortunately, the color and font information can be obtained in the GUI code in the runtime for power estimation.

6.3 Experimental Results

We have implemented the object-based power estimation on .NET Compact Framework and C#, for Windows Mobile-based mobile embedded systems. In C#, most GUI objects belong to namespace System.Windows.Controls. All objects provide height, width, and background color, which are enough for power estimation. We use eight sample programs with 10 GUIs from the Windows Mobile 5.0 SDK R2 to evaluate the implementation. Using code-level model, we estimate the power consumption of the 10 GUIs, and compare the results with the estimation from the pixel-level model and measurement. Figure 7 presents the results and shows that our code-level model with text achieves higher than 95% accuracy for all the benchmark GUIs.

7. Conclusions

In this work, we provided models for efficient and accurate power estimation for OLED displays, at pixel, image, and code levels, respectively. The pixel-level model built from measurements achieves 99% accuracy in power estimation for 300 benchmark

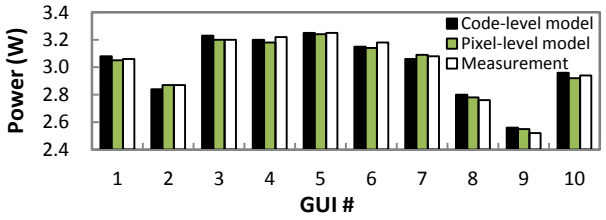

Figure 7. Power estimation comparison

images. By aggregating the power of a small group of pixels instead of all the pixels in an image, our image-level power model reduces the computation cost by 1600 times, while achieving 90% accuracy in power estimation. Our code-level model utilizes specification of the GUI objects to calculate the power consumption, which guarantees 95% accuracy.

The three power models we presented require different inputs and provide different tradeoffs between accuracy and efficiency. Therefore, we intend them for implementation and use at different system components. The pixel and image-level models are best for implementation in hardware or the operating system because they need to access the frame buffer. In contrast, the code-level model is best for implementation in application software or GUI development kits due to its dependence on knowing the composition of GUI object. All three models can provide power estimation for the system to better manage and optimize energy consumption, for the end user to make better tradeoffs between usability and battery lifetime, and for GUI designers to design energy-efficient GUIs.

8. ACKNOWLEDGMENTS

The work is supported in part by NSF Award IIS/HCC 0713249 and by the Texas Instruments Leadership University program. The authors would like to thank the anonymous reviewers whose comments helped improve the final version of this paper.

9. REFERENCES

[1] F. Gatti, A. Acquaviva, L. Benini, and B. Ricco, "Low Power Control Techniques For TFT LCD Displays," in *Proc. Int. Conf. Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, Grenoble, France, 2002.

[2] W.-C. Cheng, Y. Hou, and M. Pedram, "Power Minimization in a Backlit TFT-LCD Display by Concurrent Brightness and Contrast Scaling," in *Proc. Conf. Design, Automation and Test in Europe (DATE)*, Paris, France, 2004.

[3] L. Zhong and N. K. Jha, "Energy efficiency of handheld computer interfaces: limits, characterization and practice," in *Proc. ACM/USENIX Int. Conf. Mobile Systems, Applications, and Services (MobiSys)*, Seattle, Washington, USA, 2005.

[4] L. Cheng, S. Mohapatra, M. E. Zarki, N. Dutt, and N. Venkatasubramanian, "Quality-based backlight optimization for video playback on handheld devices," *Advanced Multimedia,* vol. 2007, pp. 4-4, 2007.

[5] S. R. Forrest, "The road to high efficiency organic light emitting devices," *Organic Electronics,* vol. 4, pp. 45-48, 2003.

[6] S. Iyer, L. Luo, R. Mayo, and P. Ranganathan, "Energy-Adaptive Display System Designs for Future Mobile Environments," in *Proc. ACM/USENIX Int. Conf. Mobile Systems, Applications, and Services (MobiSys)* San Francisco, California, USA, 2003.

[7] C. A. Poynton, *Digital Video and HDTV: Algorithms and Interfaces*. San Francisco: Morgan Kaufmann, 2003.

[8] J. Shinar, *Organic Light-Emitting Devices: A Survey*: Springer, 2004.

[9] T. Harter, S. Vroegindeweij, E. Geelhoed, M. Manahan, and P. Ranganathan, "Energy-aware user interfaces: an evaluation of user acceptance," in *Proc. ACM Conf Human Factors in Computing Systems (CHI)*, Vienna, Austria, 2004.

[10] P. Ranganathan, E. Geelhoed, M. Manahan, and K. Nicholas, "Energy-Aware User Interfaces and Energy-Adaptive Displays," *IEEE COMPUTER,* pp. 31-38, 2006.

[11] N. Kamijoh, T. Inoue, C. M. Olsen, M. T. Raghunath, and C. Narayanaswami, "Energy trade-offs in the IBM wristwatch computer," in *Proc. IEEE Int. Sym. Wearable Computers*, Zurich, Switzerland, 2001.

[12] I. Choi, H. Shim, and N. Chang, "Low-power color TFT LCD display for hand-held embedded systems," in *Proc. Int. Sym. Low Power Electronics and Design (ILSPED)*, Monterey, California, USA, 2002.

[13] N. Chang, I. Choi, and H. Shim, "DLS: dynamic backlight luminance scaling of liquid crystal display," *IEEE Trans. Very Large Scale Integration (VLSI) Systems,* vol. 12, pp. 837-846, 2004.

[14] H. Shim, N. Chang, and M. Pedram, "A Backlight Power Management Framework for Battery-Operated Multimedia Systems," *IEEE Design & Test,* vol. 21, pp. 388-396, 2004.

[15] A. K. Bhowmik and R. J. Brennan, "System-Level Display Power Reduction Technologies for Portable Computing and Communications Devices," in *Proc. IEEE Int. Conf. Portable Information Devices*, Orlando, Florida , USA, 2007.

[16] A. Iranli, H. Fatemi, and M. Pedram, "HEBS: histogram equalization for backlight scaling," in *Proc. Conf. Design, Automation and Test in Europe (DATE)*, 2005, pp. 346-351 Vol. 1.

[17] W.-C. Cheng and C.-F. Chao, "Minimization for LED-backlit TFT-LCDs," in *Proc. ACM/IEEE Design Automation Conf. (DAC)*, San Francisco, California, USA, 2006.

[18] A. Iranli and M. Pedram, "DTM: dynamic tone mapping for backlight scaling," in *Proc. ACM/IEEE Design Automation Conf.(DAC)*, Anaheim, California, USA, 2005.

[19] C.-N. Wu and W.-C. Cheng, "Viewing direction-aware backlight scaling," in *Proc. ACM Great Lakes Sym. VLSI (GLVLSI)*, Stresa-Lago Maggiore, Italy, 2007.

[20] W.-C. Cheng and C.-F. Chao, "Perception-guided power minimization for color sequential displays," in *Proc. ACM Great Lakes Sym. VLSI (GLVLSI)*, Philadelphia, PA, USA, 2006.

[21] W.-C. Cheng, C.-F. Hsu, and C.-F. Chao, "Temporal vision-guided energy minimization for portable displays," in *Proc. Int. Sym. Low Power Electronics and Design (ISLPED)*, Tegernsee, Bavaria, Germany, 2006.

[22] S. Salerno, A. Bocca, E. Macii, and M. Poncino, "Limited intra-word transition codes: an energy-efficient bus encoding for LCD display interfaces," in *Proc. Int. Sym. Low Power Electronics and Design (ISLPED)*, Newport Beach, California, USA, 2004.

[23] H. Shim, N. Chang, and M. Pedram, "A compressed frame buffer to reduce display power consumption in mobile systems," in *Proc. IEEE Conf. Asia South Pacific Design Automation (ASPDAC)*, Yokohama, Japan, 2004.

[24] 4D Systems，http://www.4dsystems.com.au/.