Energy-Efficient Graphical User Interface Design

Keith S. Vallerio *Member, IEEE,*, Lin Zhong *Student Member, IEEE,* and Niraj K. Jha *Fellow, IEEE,*

***Abstract*— Mobile computers, such as cell phones and personal digital assistants (PDAs), have dramatically increased in sophistication. At the same time, the desire of consumers for portability limits battery size. As a result, many researchers have targeted hardware and software energy optimization. However, most of these techniques focus on compute-intensive applications, rather than interactive applications, which are dominant in mobile computers. These systems frequently use graphical user interfaces (GUIs) to handle human-computer interaction. This paper is the first to explore how GUIs can be designed to improve system energy efficiency. We investigate how GUI design approaches should be changed to improve system energy efficiency and provide specific suggestions to mobile computer designers to enable them to develop more energy-efficient systems. We demonstrate that energy-efficient GUI (E^2GUI) design techniques can improve the average system energy of three benchmarks (text-viewer, personnel viewer, and calculator) by 26.9%, 45.2% and 16.4%, respectively. Average performance is simultaneously improved by 23.7%, 34.6% and 19.3%, respectively.**

***Index Terms*— Graphical user interfaces, Handheld computers, Low power.**

I. Introduction

Modern mobile computers have become an integral part of the lives of millions of users. The increasing computing and storage capacities have enabled them to provide many valuable services to users. However, reducing energy consumption is still a major design challenge. As a result, much research has been directed toward developing energy-saving techniques from hardware, software, and system perspectives [1]–[3]. Although such techniques have been successful for compute-intensive systems, many applications for modern mobile computers involve a significant amount of user interaction, which is usually handled by GUIs. We have characterized the energy consumption of different GUI features and noted the potential impact of GUIs on energy efficiency in [4]. Based on the energy characterization of different user interface technologies on handheld computers, we have pointed out that energy efficiency of interactive systems is really the product of power efficiency and user productivity in [5]. In this work, we propose to investigate the energy efficiency problem from a fourth perspective, the user's perspective. Specifically, we present techniques for E^2GUI design, reducing system energy consumption through optimization of human-computer interaction.

There are some related works. In [6]–[9], the authors proposed techniques to adjust display resolution, contrast and brightness based on human factors to reduce display power consumption. Researchers at HP Labs have investigated how future organic light-emitting diodes (OLED) based displays can be darkened to reduce power for a given GUI [10]. They have also studied how users may accept this power-saving technique in [11], [12]. In [13], we studied how user interface information can be used to predict user delays, using which aggressive system power management can be performed. All these works are hardware-oriented. They reduce power consumption through hardware techniques such as processor power management and display darkening, given the application or human-computer interaction. Their approaches are orthogonal to those described in this work, which reduce system energy consumption by optimizing human-computer interaction and GUI designs.

In this paper, we make the following contributions.

- We demonstrate the impact of GUI design on mobile computer energy efficiency.
- We propose and examine GUI design techniques for improving mobile computer energy efficiency.
- We provide specific suggestions to GUI designers:
  - remove vestiges of desktop systems (i.e., scrollbars, extraneous animations, etc.) and
  - make full use of available screen space (i.e., button size and content placement).

To the best of our knowledge, this work is the first to explore how GUIs can be designed to improve system energy efficiency.

The paper is organized as follows. In Section II, we offer background information and discuss opportunities for designing E^2GUIs. In Section III, we present specific techniques for improving system energy efficiency. After that, we detail our experimental setup for both participants and hardware in Section IV and present experimental results in Section V. Section VI discusses the experimental results and is followed by a survey of mobile computer users in Section VII. The final conclusions of this work are presented in Section VIII.

II. Background

This section provides background information related to mobile computer energy consumption. We first define system energy efficiency for mobile computers and then address user's impact on it.

A. System energy efficiency

Many modern mobile computers are interactive. For these systems, power consumption is an insufficient metric for gauging energy efficiency since execution times may vary significantly. Since most of these systems are goal-driven, we define a *task* to be the set of user actions and system

Manuscript received June 10, 2004; revised March 9, 2005. This work was supported by the DARPA under contract no. DAAB07-02-C-P302.

Keith S. Vallerio is with ALK Technologies, Inc. Lin Zhong and Niraj K. Jha are with the Department of Electrical Engineering, Princeton University, NJ 08544.

responses required to achieve a meaningful goal, such as adding information on a contact or viewing a schedule. The energy consumed by the system between the beginning and end of these actions is called the task energy, or *energy per task*, which is used as the measure for energy efficiency in this study. Energy per task includes the energy consumed during both system busy and wait times. In [5], we define system energy efficiency as user productivity times power efficiency, which is essentially the same as energy per task.

B. User's impact

As we have discussed in [4], different GUI features have different system energy impacts. Different users may utilize different GUI features to finish the same task, leading to different levels of system energy efficiency. Therefore, providing users with energy-efficient GUI features and training or limiting them to use these features will improve energy efficiency as we will see later.

More importantly, users are slow relative to computers. Our analysis has shown that interactive mobile computers spend most of their time in idle periods, waiting for user responses [13]. Displays, well known as one of the largest single power-consumers in mobile computers [4], [14]–[16], usually have to be on in these idle periods. As a result, interactive mobile computers also spend most of their energy waiting for user responses. Therefore, improving user productivity is extremely important for improving system energy efficiency.

The user response time for applications depends on the design of their GUIs. According to the *model human processor* [17], three basic processes are involved in this interaction: perceptual capacity, cognitive speed and motor speed. Each of these processes is explained next.

1) Perceptual capacity: Psychological studies [18] have shown that humans read through discrete eye movements, which consist of a series of fixations which are separated by saccades. During fixations, the eye stares at a particular point of interest until the next saccade, which is a quick movement to the next fixation. For reading text, the average fixation duration is 240ms and is affected by word layout [19]. Only recently have formal studies been performed involving both text and images [20]. Most significantly, it was found that people frequently read text titles before viewing images that accompany the text. To our knowledge, no work has been performed regarding eye movement patterns in the presence of text and GUI widgets (e.g., pushbuttons, menu bars, etc.). This work does not investigate the effects of GUI placement on reading speed, but it is clear from previous work with text and images that proper selection of content placement can improve GUI interaction speed.

Another factor that influences reading speed is the visibility of the material being presented. Visibility, in turn, depends on the color scheme, contrast ratio and luminance [21]. Human vision has different sensitivities to different colors. GUIs with better color schemes and contrast ratios are easier to read.

2) Cognitive speed: The Hick-Hyman Law [22], [23] states that the reaction time, RT, required to make a decision based on N distinct and equally possible choices is given by the following equation:

$$RT = a + b \cdot log_2 N$$

where a and b are constants. The insight from this law is that to accelerate the human cognitive process, a GUI should present as few choices as possible. Split menus, presented in [24], achieve this goal by separating out the most common functionality into a smaller menu. Users are provided a mechanism to turn on advanced features, which causes the less common menu items to be displayed. By reducing the number of options users are presented with in the most frequent case, split menus capitalize on the Hick-Hyman Law.

3) Motor speed: The motor speed of human users is governed by the Fitts Law [25]. This relationship is commonly presented as the following equation:

$$T = c_1 + c_2 \cdot log_2(\frac{D}{W} + 1)$$

where T is the time required to complete a task (i.e., moving from point A to point B), D is the distance between A and B, W is the width of the target and c_1 and c_2 are experimentally-determined constants. Based on this law, a GUI should utilize as much screen area as possible for widgets to be hit. Widgets that are supposed to be hit sequentially should be placed near each other. Much effort has been spent in exploiting this law to improve user speed [26]–[29]. We will present new energy-efficiency techniques based on this law.

III. GUI DESIGN TECHNIQUES FOR ENERGY REDUCTION

This section presents a framework and specific techniques for designing E^2GUIs for mobile computers. An overview of E^2GUI design opportunities and methods is presented first. This is followed by a discussion of GUI categorization. Finally, a detailed description of the E^2GUI design techniques is presented.

A. Overview

Developing a framework for designing E^2GUIs for mobile computers requires an understanding of how these GUIs are used. Most mobile computers utilize relatively small displays in order to achieve higher portability. GUI design for these small displays requires a different approach from traditional GUI design. However, this fact is not well-recognized. Many GUIs on handheld computers appear to be miniature versions of their desktop counterparts. These GUIs contain power-hungry widgets that are vestiges of desktop GUI design. The scrollbar is a good example of this type of widget. Although commonly used on desktop systems, the frequent screen updates it generates consume more power than a pair of up and down buttons [4].

GUIs for mobile computers are often designed without considering the impact of using smaller displays. Thus, mobile computer energy efficiency can be improved by better utilizing the limited screen real estate. Although most users are fairly skilled at using a keyboard and mouse with desktop/laptop computers, they are not equally good at using a stylus, touch-screen, or virtual keyboard, which are commonly found on

modern handheld computers. Slow input time is one of the most important factors limiting user productivity and increasing the energy consumption of these systems.

The first step in the design process is to categorize GUIs based on the primary interaction between a system and its user. After a GUI has been categorized, an E^2GUI design is tailored to the application type by applying the appropriate optimization techniques. The optimization techniques are divided into power reduction techniques, performance enhancement techniques and facilitators. In particular, this paper introduces the following techniques:

- Power reduction
  - Low-energy color scheme
  - Reduced screen changes
- Performance enhancement
  - Hot keys
  - User input caches
  - content placement
- Facilitators
  - Paged display
  - Quick buttons

A detailed explanation of each of these techniques is presented in Section III-C.

B. Categorization

GUIs for PDAs can be broadly divided into three categories based on their purpose: input-centric, content-centric, and hybrid. The primary action performed by input-centric GUIs is obtaining user input. These GUIs include text messaging, calculators, and terminal sessions. Input-centric GUIs can be further subdivided based on the complexity of their input mechanisms. Content-centric GUIs are mostly used for viewing stored data. Examples include map software, text viewers and web browsers. Hybrid GUIs require noticeable user input while dedicating a significant portion of the screen to displaying data. Examples are text editors, address books, and configuration menus. Several common GUIs are categorized in Fig. 1.

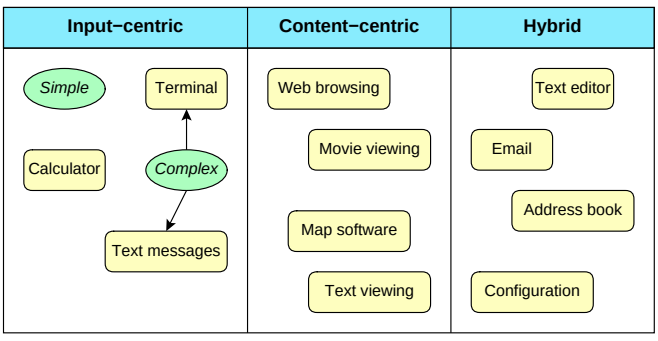


Fig. 1. Categorization of several common PDA GUIs

In categorizing GUIs, it becomes evident that the needs of input-centric and content-centric GUIs differ noticeably. Users spend much more effort generating input for input-centric GUIs than they do for content-centric. Thus, the former GUI type should be designed for ease of input, and the latter for ease of browsing.

C. Optimization techniques

As mentioned before, E^2GUI design is divided into power reduction techniques, performance enhancing techniques, and facilitators. Power reduction techniques reduce the power consumed by the display subsystem. Performance enhancing techniques reduce energy by improving user interaction speed. Facilitators do not reduce energy directly. Instead, they enable other techniques to be used more effectively. A summary of E^2GUI design techniques is presented in Table I. The names of the techniques are listed in Column 1. Columns 2–4 indicate the category of GUI that can benefit from each technique (input-centric, content-centric, hybrid). Finally, Columns 5–7 indicate the technique's type.

1) Power reduction: GUIs are an important target for power reduction. Currently, most displays used for mobile computers are based on the liquid-crystal display (LCD) technology. For LCDs and future OLED-based displays, different colors, color patterns or color sequences consume different amounts of power, which can be exploited for energy efficiency [4], [30]. A description of the contribution of individual GUI features to mobile computer energy consumption is presented in [4].

The first power reduction technique, *low-energy color schemes*, reduces display energy by using colors and color patterns that consume less power. For example, thin film transistor (TFT) LCDs, which are typically used in PDAs, consume more power when white than when black [4]. For future OLED-based displays, display power consumption will be proportional to the number of on pixels and their luminance [10], [30]. Researchers from HP Labs have investigated how OLED-based displays can be partially darkened to reduce power consumption [10]. Moreover, power consumption can be reduced by selecting energy-efficient colors for GUIs. Of course, it is important to take visibility (color scheme, contrast ratio, and luminance) into account. Yellow may require less power than black, but a yellow-on-white display is much more difficult for a user to read than a black-on-white display. In addition to selecting energy-efficient colors, fine patterns and textures should be avoided, since they increase power consumption by increasing switching activity.

The second power reduction technique, *reduced screen changes*, reduces energy by reducing the switching activity as well as computation required for screen data generation. Many of the opportunities for reducing extraneous screen changes result from GUIs being designed for aesthetic purposes rather than energy efficiency. This can be seen in the following examples.

- User-perceived responsiveness is different from real responsiveness. Using a progress bar may make a user feel that the computer is more responsive, but it actually slows down the system and increases energy consumption.
- Animations give users a natural feeling for screen changes ($e.g.$, animations for pop-up menu windows, closing windows, and minimizing windows). However, these animations waste energy. They add little or no functionality, but they increase the power consumption.

TABLE I
E^2GUI DESIGN OPTIMIZATIONS

Technique	Input	Content	Hybrid	Power	Performance	Facilitator
Low-energy color scheme	x	x	x	x		
Reduced screen changes	x	x	x	x		
Hot keys	x	x	x		x	
User input cache	x		x		x	
Content placement	x	x	x		x	
Paged display	x		x			x
Quick buttons	x		x			x

- Many natural-to-use GUI widgets, such as scrollbars, were designed based on desktop GUIs and ported to handheld GUIs. However, scrollbars are very energy-inefficient since they cause frequent screen updates, leading to much more power consumption [4].

These modifications can be made to several GUIs that are common on modern PDAs. Most notably, the results listed in Section V indicate that replacing the scrollbar with up and down buttons can significantly improve energy efficiency. It is worth noting that the system power impact of color and screen changes comes from not only the display panel but also the display controller and even the processor.

2) Performance enhancement: Performance enhancement techniques focus on improving user productivity. Note that traditional software energy optimization techniques are often also based on reducing run-times through streamlining of code. Performance enhancements from better GUI design are more significant for interactive systems since user input usually requires orders of magnitude more time than computation. As mentioned previously, interactive systems usually spend most of their time and energy in waiting for user responses.

Hot keys enable traditional user inputs (save, open, close, cut, paste, etc.) via multi-key gestures (i.e., control-s, control-o, etc.). This technique is commonly used to increase user interaction performance on desktops but is not frequently used in mobile computers, such as PDAs, since the use of a stylus for input decreases the advantages of multi-key gestures. However, this energy reduction technique becomes viable through the use of the quick buttons facilitator (see Section III-C.3) since the quick buttons enable the operation to be performed in a single gesture.

User input caches store the most recent (or most frequent) inputs by users. The small size of PDAs makes sizable text inputs tedious and time-consuming. By caching previous inputs, the input time can be greatly reduced. This method is used in the calculator benchmark described later on. It can also be combined with paged displays (see Section III-C.3). The user can switch to another page (displaying the input cache entries) and quickly select the input to be reproduced. The input cache entries can be based on most recent use, most frequent use, or most common use. User input caches are most effective for user input intensive programs and can also benefit from quick buttons (see Section III-C.3). This technique is used in the calculator benchmark described in Section IV-B.3. In general, user input caches are most useful when a small set of known inputs occurs frequently. The autocompletion mechanism, widely used in virtual keyboards, is a good example of this. However, the results of our study indicate that the cache hit must save a critical number of keystrokes before it is worthwhile. Thus, autocompletion functions would be more energy-efficient if they left out the completions consisting of only one or two additional letters.

Another use of user input caches is to store commonly used phrases for text messaging. In this case, text messages that are frequently used by a user (or generic common phrases) should be stored in user input caches. This enables longer strings of text to be entered with fewer keystrokes, thus requiring less time. Text messaging, primarily available only on mobile phones currently, is a rapidly growing medium for communication and will become more essential once other mobile computers are able to communicate wirelessly on a regular basis.

Content placement reduces the user interaction time for frequent inputs by strategically laying out the GUI content. There are three major considerations when preparing a GUI layout: perception capacity, motor speed, and cognitive speed.

Although eye movements including text and GUI objects have yet to be studied, psychological studies demonstrate that users have innate tendencies regarding where they focus their attention in the presence of both text and graphic material [20]. In particular, their eyes are drawn toward large text first. Afterwards, many will focus on a picture and then on the remaining smaller text. Each eye focus requires hundreds of milliseconds, which is significant compared to the number of calculations that can be performed by a mobile computer in the same timespan. Thus, perception latency can be reduced by decreasing the amount of eye movements the user must make. This technique is used in the text-reader benchmark described in Section IV-B.1. The E^2GUI uses the whole screen for up and down buttons, so the user does not have to focus on the right hand side of the screen to locate the scrollbar.

Content placement can also reduce user interaction time by using large fonts and pictures to draw user attention to important objects quicker. A final point in GUI design is that readers sometimes quit reading although they have not read the whole text. Thus, breaking up text into manageable regions could promote faster comprehension and interaction. However, these last two points are beyond the scope of this work.

The second consideration, reducing motor latency, is based on the Fitts Law. Frequently-pressed buttons should be as large

as possible and located near each other. An example is the use of the whole screen to control page scrolling instead of small buttons (or the scrollbar) on one side. This method is used in the text-reader benchmark described in Section IV-B.1 and the personnel viewer benchmark described in Section IV-B.2. A natural consequence of this design rule is that tasks that require multiple interactions should be simplified into ones requiring fewer interactions whenever possible. This method is used in the personnel viewer benchmark.

The third consideration, reducing cognition latency, can be achieved by decreasing the number of options users have to select from. Split menus are a good example of using content placement to reduce cognition latency. Although user input caches violate this design guideline since they add to the number of options a user is presented with, the results in Section V indicate that the savings achieved outweigh the cognitive and motor speed penalties incurred (as long as the cache entries are sufficiently long).

3) Facilitators: The facilitators are a pair of techniques that enable or enhance the effects of the other techniques. The first facilitator, *paged display*, enables increased user interface functionality by increasing the effective display size. It is similar to tabbed panels. Spreadsheet programs currently utilize this technique by dividing sessions into worksheets. This technique can be utilized to include a page of buttons designed to enhance the user interaction speed. This allows the buttons to be larger, which can reduce power consumption, as illustrated by the results presented in Section V-C.

The second facilitator, *quick buttons*, uses available hardware buttons to increase user interaction capabilities for the current application. Holding down a hardware button can act as a <SHIFT> or <CTRL> key, which can facilitate the use of key combinations that are traditionally impractical for handhelds. (Note that one straightforward use for quick buttons is to enable rapid switching between paged displays.)

IV. Experiments

This section describes the experimental setup and GUI applications used to validate this work. The experiments were performed using the original version and an energy-efficient version of three GUI applications. Thus, six programs were involved: two implementations for each of the three applications. Although the three applications are not benchmarks in the strictest sense, the term benchmarks is used to refer to a pair of implementations (the original and the energy-efficient version of the application). One set of experiments was performed for each implementation of a benchmark.

A. Experimental setup

We first describe the participants, experimental procedure and hardware devices used for the experiments.

1) Participants: A total of 16 people participated in the three sets of experiments. There were 10 male and six female graduate students majoring in electrical engineering and computer science. For each benchmark, a different set of six participants was selected. Only two of them participated in experiments for a second benchmark.

2) Experimental procedure: For each version of a benchmark, an experiment set for a participant consisted of a training session followed by a series of experiments (runs). In the training session, the functionality of the given version of the benchmark was demonstrated to the participant. The participant was then instructed to perform practice tasks that were similar to the ones used in the experiments later. The practice tasks acquainted the participant with the version of the benchmark and with the task being performed. Once the participant felt comfortable performing the training tasks, the experiments were performed using different data than the training version of the benchmark. This procedure was then repeated for the other version of the GUI. Half of the participants started with the original GUI and half started with the E^2GUI to make the comparison fair.

The calculator benchmark was an exception to this procedure. Participants were first trained with both versions. Then eight runs for the two versions, four for each, were interleaved.

In each run, the participants were instructed to say "Go" when they were ready to start and "Done" when they had completed the task. Time and power were measured from the time "Go" was uttered until the time "Done" was uttered. Since each run of experiments usually lasted more than tens of seconds, any experimental error incurred through the use of voice commands did not have a significant impact on the results of the experiments. In some experiments, such as those for the text-reader benchmark, it is impossible for the computer to determine whether a participant has started or has found the target line of text. This is another reason that we asked participants to use voice signals to indicate when they started and got done. The data used for the experiments were designed to be easily memorized, so that the participants would remember the data before the experiments were conducted. The data were similar to the data used in the training tasks, but with slight differences. For example, the text reader used a different story for the actual experiment and each run consisted of looking up a different sentence. To minimize distractions, talking during and between experiments was discouraged.

3) Hardware: The PDA used in our experiments was a Sharp Zaurus SL-5500 [31] running Embedix Linux and Qtopia. It is equipped with a 206MHz StrongARM SA-1110 processor, 64MB SDRAM and 16MB FLASH ROM. It has a 3.5" 240x320 reflective TFT LCD with 16-bit color. The experiments were performed in a well-lit office. Hence, the front light was turned off. The benchmarks were based on the Qt/Embedded GUI platform.

Power measurements were made using an Agilent 34401A digital multimeter connected to a PC running Windows. Power was determined by measuring current through a 0.1Ω sense resistor connected in series in the DC power supply cord to the PDA. A program running on the PC instructed the multimeter to sample the voltage drop cross the resistor at 200Hz and report the average value at 8Hz. The program converted the voltage drop into system power consumption. Energy for each task was determined by integrating the power samples from the start to the completion of the experiment.

B. Benchmarks

Three GUI benchmarks were used to examine and verify different E^2GUI design techniques. Here, we describe their functionality, implementation, the tasks to be performed, and what techniques they are intended to demonstrate.

1) Text-reader with screen buttons: We first modified the text editor included in Qtopia-1.5.0 [32] for a text-reader as the first benchmark. The original version of the text-reader looks and functions the same way as a standard text viewer and a user browses the text with the scrollbar. To create an energy-efficient text-reader, we made the entire screen act as a pair of up and down buttons, called *screen buttons*. Thus, a user can browse by touching the screen anywhere in its top half and scroll down by touching it anywhere in its bottom half, instead of using the scrollbar. Fig. 2 presents a single image of the text-reader since both versions look identical. Note that the divider line depicted in the figure is not part of the GUI, but is there to illustrate the division between the two screen buttons.

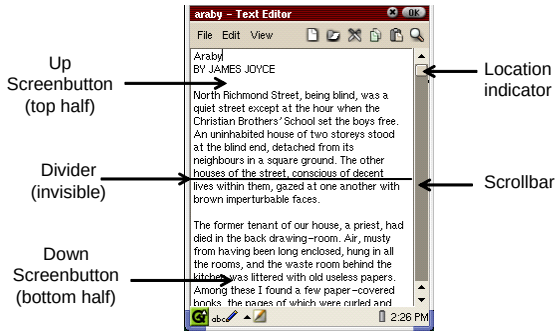


Fig. 2. Text-reader benchmark

The purpose of this benchmark was to examine the efficacy of using part of a GUI that is usually read-only for obtaining inputs as well. Using traditionally non-interactive regions of the screen (the text display portion) better utilizes screen real estate. This benchmark is an example of content placement and reducing screen changes.

For this benchmark, two runs were performed on each of the original and energy-efficient implementations for each participant. In each run, a participant was given a line of text to look up within a short story. The participants were told that the text would be positioned at the beginning of a paragraph so that the experiment could be focused on the browsing capability of the two versions of the GUI. Thus, the effects of reading speed could be kept to a minimum.

2) Personnel viewer with multiple techniques: The second benchmark was a personnel viewer application. The personnel viewer allows a user to scroll through a list of names and display information about selected individuals. Radio buttons at the top allow a user to select between position, affiliation, and health queries. In this example, a straightforward implementation is compared with an energy-efficient implementation. Fig. 3 depicts the main screen for the two implementations. The original GUI requires two button presses to display information of the selected individual: the radio button at the top to select the data to be displayed and the pushbutton on the bottom to process the selection. The energy-efficient version replaces this two-button combination with a single set of pushbuttons on the top. The energy-efficient version also replaces the scrollbar with up and down buttons and alphabetic index tabs. Finally, the energy-efficient version has a low-power color scheme.

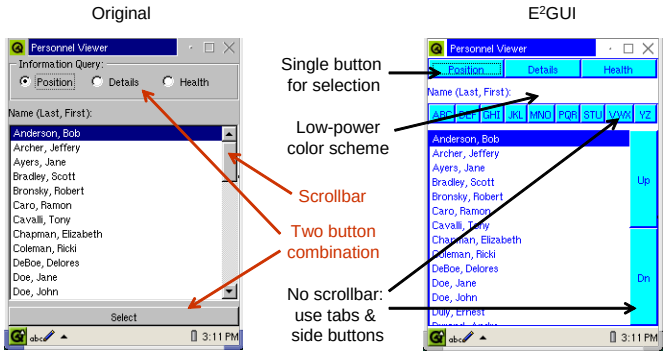


Fig. 3. Personnel viewer benchmark

There were two main purposes for choosing this benchmark. The first was to demonstrate the impact energy-efficient design techniques have on more complex tasks. The second was to determine the effect of using a combination of energy-efficient design techniques. This benchmark was designed to test the combination: content placement, reduced screen changes (removing the scrollbar) and low-energy color schemes.

For this benchmark, two runs were performed on each of the original and energy-efficient implementations for each participant. In each run, the participant was given a name to look up and a query selection to make.

3) Calculator with input caching: The third benchmark was based on the calculator included in Qtopia-1.5.0 [32]. The original version of the calculator looks and functions in the same way as a real calculator. To create an energy-efficient calculator, we implemented an input cache button on each of the digit buttons. The input cache buttons store the most recent user input that begins with that digit. Fig. 4 depicts the two implementations of the calculator. It can be seen from the figure that a small portion of each digit button is designated as an input cache button in the energy-efficient version. For example, since the last value entered was 512, the input cache button on the 5 button displays 12 to indicate that pressing that button will enter 512. The energy-efficient calculator also makes use of a low-power color scheme.

The purpose of this benchmark was to explore the strength and weakness of using input caching. Adding a user input cache enables users to perform multiple inputs with a single gesture, provided that the desired inputs are contained in the cache. However, the benefits of a user input cache come with a price. Adding a user input cache introduces a penalty for cache misses. The user input cache buttons occupy valuable screen space, resulting in smaller sizes for the remaining GUI elements. Thus, use of the GUI will require more time for a GUI with a user input cache if the cache cannot be used. (Note that this effect can be mitigated through the use of

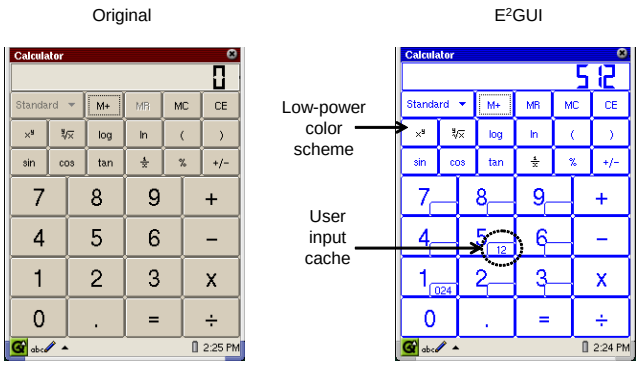


Fig. 4. Calculator benchmark

the facilitators mentioned in Section III-C.3.) This experiment was designed to investigate the relationship between user interaction time and user input cache size.

A set of four experiments were performed on each of the original and energy-efficient implementations. In each experiment, the participants calculated the product of a series of numbers, which had a repetitive pattern to make them easier to remember.

V. Experimental results

In this section, we present the results of the experiments on the three benchmarks. The results are reported as average improvements in performance and energy consumption. For each of the two versions of a benchmark, there are N runs of experiments for a participant. N is different for different benchmarks as described in the previous section. Let T_i denote the task duration for the ith run of a participant with the original version, and T_i' denote that for the ith run with the energy-efficient version. The average performance improvement for the participant with the benchmark is calculated as

$$\text{avg. perf. improvement} = 100\% \cdot \frac{1}{N} \cdot \sum_{i=1}^{N} \frac{(T_i - T_i')}{T_i}$$

The average energy improvement is calculated in a similar fashion. Note that the ith run of the original version can be either before or after the ith run of the energy-efficient version for a participant. It was randomly decided for each participant.

A. Text-reader with screen buttons

As mentioned in Section IV-B.1, the set of experiments for this benchmark consisted of two trials for each GUI. Fig. 5 depicts the average performance and energy improvement for the E^2GUI. The X axis indicates the participant number and the Y axis indicates the percent improvement due to E^2GUI. The left bar indicates improvement in performance and the right bar indicates energy savings.

From the figure, it can be seen that most of the participants benefit from the E^2GUI. However, it can be seen from the figure that improvements varied a lot from participant to participant. One participant ($i.e.$, the fourth one) did not even benefit from this technique. There is a similar variance for the other two benchmarks as will be seen later. Such a variance in improvements can be attributed to the following reasons. First, different participants adopted very different strategies for tackling the assigned tasks. Thus, their performances differed significantly, which made the corresponding percentage contributions of E^2GUI different. The average performance and energy consumption for the original version are presented in Fig. 6, which shows a large variance among participants. Second, the participants had different prior experience with GUI operations. For example, they differed in their skills in using the standard method of scrolling through a text box, leading to different improvements for the E^2GUI. A third possible reason was that participants differed in how effective they were in learning how to use new GUIs. For example, the fourth participant might not have had sufficient time to become acquainted with the new GUI. Variance among participants is really due to the inherent differences among individuals. Nevertheless, the results are still encouraging since most users realized rather dramatic improvements in performance and energy savings after only a few minutes of practice with the new GUI.

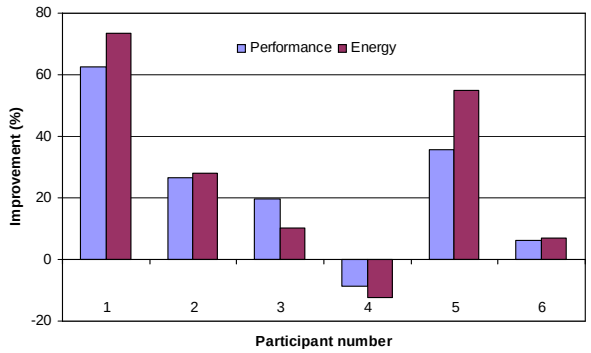


Fig. 5. Text-reader benchmark: Average performance and energy improvements

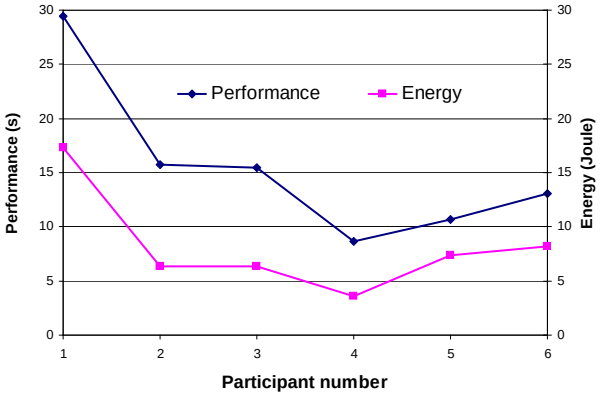


Fig. 6. Text-reader benchmark: Average performance and energy consumption for the original version

The average improvement in performance for the text-reader is 23.7% and the average system energy saving is 26.9%. The improvements in energy and performance for this GUI come from replacing the scrollbar with the screen buttons. Screen buttons are energy-efficient and convenient since they are large

and do not require users to move their eyes away from the text.

B. Personnel viewer with multiple techniques

As mentioned in Section IV-B.2, the set of experiments for this benchmark consisted of two trials for each GUI. Fig. 7 depicts the average performance and energy improvement for the E^2GUI. The X axis indicates the participant number and the Y axis indicates the percent improvement due to the E^2GUI. Similar to Fig. 5, the left bar indicates improvement in performance and the right bar indicates energy savings. From the figure, it can be seen that all the participants benefit from the E^2GUI. The variation in performance improvement among the participants most likely occurs for the same reasons discussed in the previous section.

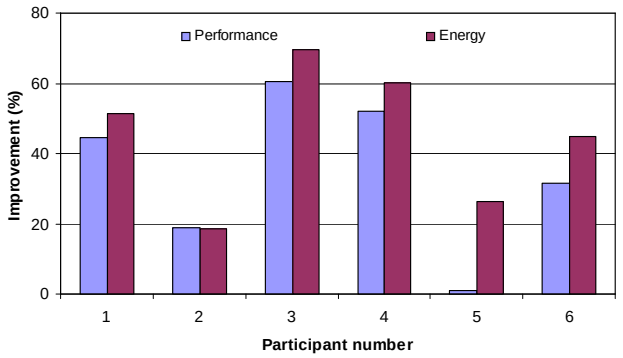


Fig. 7. Personnel viewer benchmark: Average performance and energy improvements

The average improvement in performance for the personnel viewer benchmark is 34.6% and the average system energy saving is 45.2%. These values are better than the results for the text-reader and demonstrate that complex GUIs can benefit from utilizing multiple energy-efficient design techniques.

The impact an E^2GUI design can have is demonstrated in Fig. 8. Fig. 8(a) corresponds to the original GUI and Fig. 8(b) to the E^2GUI. The X and Y axes of each chart indicate time and power consumed by the application on the PDA in seconds and Watts, respectively. Notice that although the range for the Y-axes are similar, the E^2GUI has considerably fewer and thinner spikes. This corresponds to less energy consumption. A second difference to notice is that the X-axis of the E^2GUI is noticeably shorter than the one for the original. The combination of lower power and better performance allows the E^2GUI to consume 25% less energy for this example.

An analysis of Fig. 8 reveals that scrollbars are highly inefficient from an energy perspective. (Note that this is true for the text-reader benchmark as well.) The power curves of the original GUIs (using the scrollbars) had more peak power spikes since tracking the stylus requires continuous computation and screen updates. Furthermore, several participants reported that it was more difficult for them to use scrollbars (with the stylus) for browsing. They preferred using the buttons for both GUIs. Although user preferences are investigated in Section VII, the results of the experiments on the personnel viewer and text-reader clearly indicate that the scrollbar is not energy-efficient. Scrollbars were designed for desktop interactions and these experiments demonstrate that this vestige of desktop systems is not that well suited to handheld GUIs.

C. Calculator with input caching

This benchmark was different from the others since it consisted of four trials for each GUI instead of two. The user input vector for each trial was a sequence of numbers that were multiplied by the participant using the calculator. The sequence included duplicate copies of each number since the experiment measured the impact of cache hits. The sequence of numbers was also designed to be easy to remember to reduce the effects cognitive processes had on the measurement. Finally, each number in a sequence had a number of digits equal to the sequence number plus one. The sequences of numbers used for the four trials are listed below:

1) $11 \times 11 \times 21 \times 21 \times \cdots \times 91 \times 91$
2) $192 \times 192 \times 292 \times 292 \times \cdots \times 992 \times 992$
3) $1927 \times 1927 \times 2927 \times 2927 \times \cdots \times 9927 \times 9927$
4) $19273 \times 19273 \times 29273 \times 29273 \times \cdots \times 99273 \times 99273$

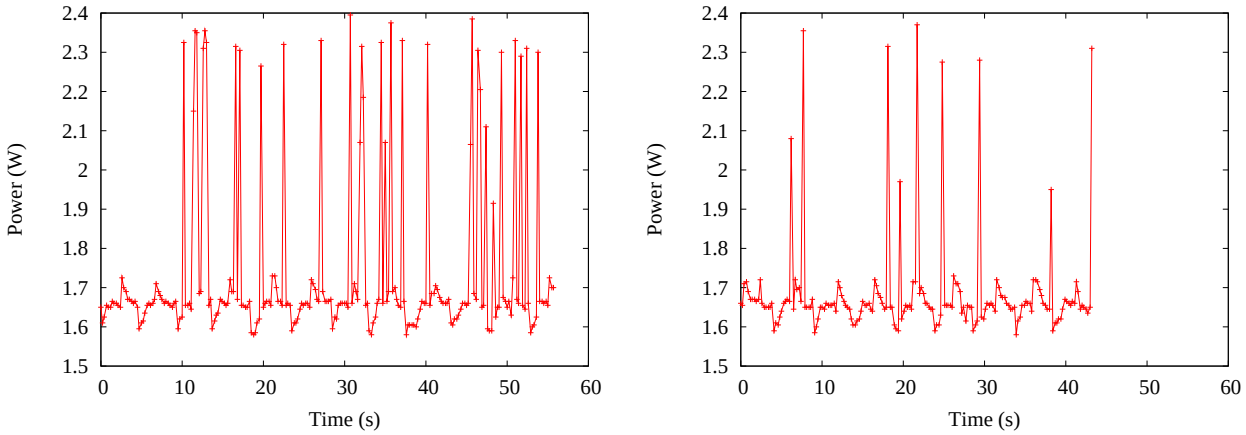
Fig. 9 depicts task energy improvements of the E^2GUI using a trend line over all four trials for each participant. The X axis indicates the trial number and the Y axis the percent improvement of the E^2GUI over the original. Pi, $1 \leq i \leq 6$, denotes the participant number. The P4 line, corresponding to the fourth participant, is the result of a user who was slower to get used to the new GUI. It is evident that the relative performance of the E^2GUI depends on the length of input data. The smaller inputs (i.e., trial 1) are more energy-consuming on the E^2GUI and the reverse is true for the larger inputs (i.e., trials 3 and 4).

Fig. 9 demonstrates the effects of adding a user input cache. Similar to cache theory for microprocessor caches, it is evident from the figure that the effectiveness of the user input cache depends on cache hits saving more time than the penalties they introduce (due to the smaller size of the digit buttons). Thus, the original is better for smaller inputs and the E^2GUI is better for larger inputs. Trial 2 shows that the user input cache benefit is completely offset by the cache miss penalty for three-digit inputs. These results are consistent with the Fitts Law and cache theory. Thus, deciding user input cache size (or if one should be used at all) can be based on the Fitts Law and the standard theory for microprocessor caches.

For trial 3, the average performance improvement is 14.2% and the average system energy saving is 13.4%. For trial 4, these values are 19.3% and 16.4%, respectively. Note that there is energy reduction only when the inputs are sufficiently long. Thus, for applications such as text messaging, the cache should not include short words. The same rule applies to the auto-completion mechanism that is widely used in a virtual keyboard.

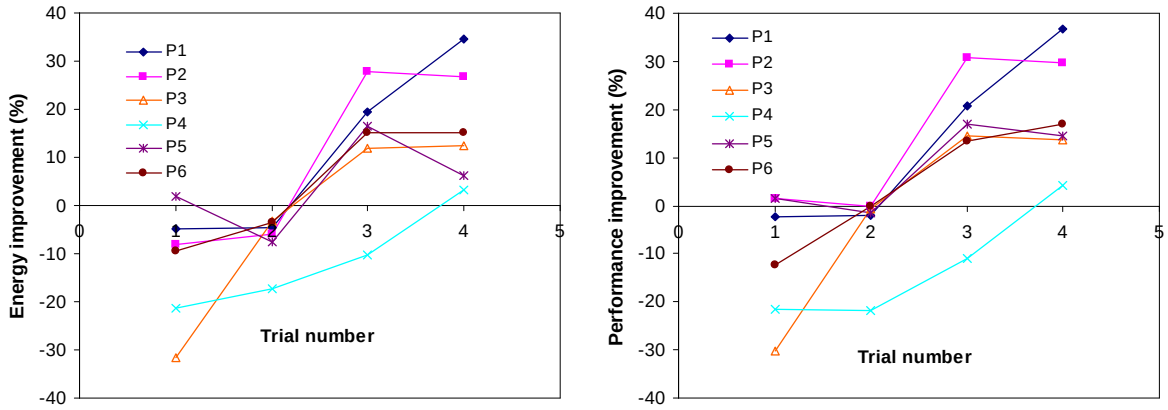
D. Stylus linger time

In comparing the performance improvements in Fig. 9(b) with the results in Fig. 9(a), it can be seen that the average



(a) Original (b) Energy-efficient

Fig. 8. Personnel viewer benchmark results



(a) Energy (b) Performance

Fig. 9. Calculator experimental results

power increased slightly for the E^2GUI. The power increase is a result of using input caches, which reduce the size of buttons. Further experiments revealed that smaller buttons are more difficult to hit and require longer *stylus linger time*, which is the amount of time the user's stylus spends in contact with the buttons. As a result, the PDA tracks the user inputs longer, and consumes more energy. Fig. 10 presents this phenomenon graphically using two excerpts for one participant. In this figure, the X axis indicates time and the Y axis power. The wider peaks in the new GUI (on top) demonstrate this effect. Note that overall system energy still went down.

In order to examine this phenomenon further, an application was designed to measure stylus linger time. The application, pictured in Fig. 11, consisted of buttons randomly numbered from one through nine. The height and width of the buttons could be scaled as well as the horizontal and vertical distances between them. Using this framework, a set of three participants, including two authors of this paper, participated in a series of experiments. In each experiment, the participant was asked to press the buttons sequentially from one to nine. The numbers displayed on the buttons were randomly ordered to make the action complex (to reduce the effects of muscle memory), yet easy for the participants to remember.

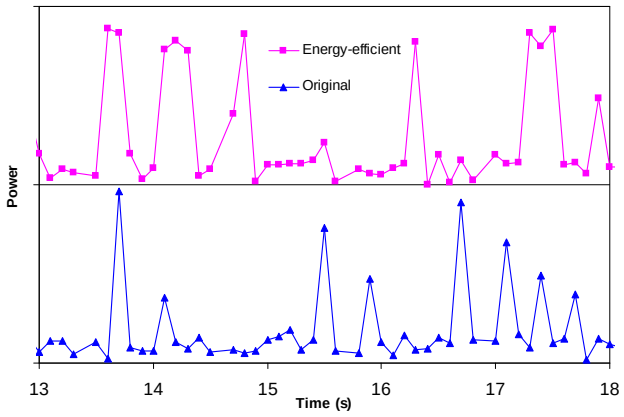


Fig. 10. Excerpts from the power data for Calculator experiments

The only measurement recorded for this set of experiments was stylus linger time. Each time the stylus comes in contact with the touch screen, a power spike is generated as the system processes the touch screen input. Stylus linger time

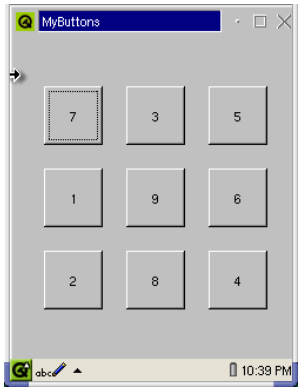


Fig. 11. Stylus linger time measurement application

was measured by determining the peak width for each button press. In order to achieve sufficient resolution, the sampling frequency of the multimeter was increased to over 200Hz. The results of this set of experiments is depicted in Fig. 12. The X axis in the figure indicates the size of the buttons in pixels and the Y axis indicates the average stylus linger time on each button. The figure demonstrates the correlation between button size and stylus linger time. Smaller buttons lead to longer linger times. A set of additional experiments, which are not included here, indicated there is a point of saturation such that larger buttons beyond a certain threshold value do not lead to shorter stylus linger times. Stylus linger time, as well as the saturation point, were found to correlate with task complexity in addition to button size. However, determining the extent of this effect was beyond the scope of this paper.

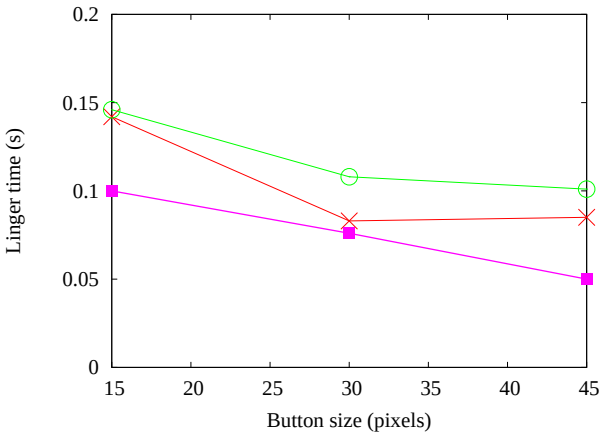


Fig. 12. Stylus linger time

VI. Discussions

The examples presented in this work utilize a variety of techniques focused on two common themes, reducing display power and increasing user interaction speed. Although energy-efficient color schemes reduce power, it is more important to avoid scrollbars and GUIs that require stylus tracking. Small buttons should also be avoided since their small size causes the user's stylus to linger longer on the touch-screen, requiring more computation energy.

The most effective way to improve energy efficiency is improving the latency caused by interfacing with humans. The simplest method for accomplishing this is to use fewer buttons and make them as large as possible. The success of this approach depends to some extent on how familiar the user is with previous interaction paradigms. When a modification is first presented, user interaction time may increase since users need time to acclimate to the new interface. The participants in our experiments greatly benefited from using the index tabs in the personnel viewer application, indicating that designing GUIs to allow the user to capitalize on context-sensitive information improves performance. Finally, user input caches are effective if they can save many keystrokes, but should not be included otherwise. For PDAs, with a limited screen space, adding a cache button consumes part of the screen and reduces GUI efficacy if there are not enough cache hits.

Based on the above observations, we make the following recommendations to GUI designers for systems with tight energy budgets.

- Reducing user interaction time should be the primary goal.
- Avoid vestiges of desktop systems (i.e., scrollbars, extraneous animations, etc.).
- Make full use of available screen space (i.e., button size and content placement).
- Cache long, frequently-used user inputs

Although the work presented in this paper focused on GUIs, this research should be helpful to the human-computer interaction (HCI) community in general. Given the diverse HCI interfaces being developed, it would be edifying to determine the relative energy efficiency of these techniques.

VII. Survey

In Section VI, a number of recommendations for GUI designers were presented. These recommendations were based on reducing system energy consumption. However, they did not consider user preferences, which play an essential role in the success of new GUI design techniques. Thus, a group of mobile computer users was surveyed to ascertain the impact of E^2GUI techniques on their mobile computing habits. The following three subsections detail the methodology used to perform the survey, the results, and conclusions that can be drawn from the survey.

A. Methodology

This section describes the participants, experimental procedure and questions used for the survey.

1) Participants: A total of 12 participants took part in the survey. The group was composed of 10 male and two female electrical engineering graduate students.

2) Experimental procedure: Each participant was told to look up a list of names using each version of the personnel viewer GUI on the PDA. (Similar to the experimental procedure, half of the participants started with the original GUI and half started with the E^2GUI to reduce the effect of

biases on the survey.) After the participants became familiar with both GUIs, they returned the PDA and picked up a questionnaire form. The procedure was complete once the participant completely answered the questions on the form.

3) Questions: The questionnaire was divided into four major sections. The first section asked the participants how frequently they used mobile computers (including PDAs, cell phones, etc.). The second and third sections of the form consisted of a series of statements that were rated by the participants. For each statement, eight options were provided ranging from strongly agree to strongly disagree, including a don't care option.

In the second section, participants were asked to rate a set of statements that compared the first GUI they used with the second GUI (GUI A vs. GUI B). In the third section, they were asked to rate a series of general statements regarding their mobile computer experiences and preferences. The final section asked specific questions regarding which GUI the participants preferred and how long they would like the battery on their mobile device to last.

B. Results

In the first section, most (75%) of the participants indicated that they used mobile computers at least once per week. This section of the questionnaire was included in case there was a large discrepancy between experienced mobile computer users and novices. However, no significant differences were found.

In the second section, a few points became apparent. First, the participants tended to believe that the first GUI they used was easier to learn. This is one of the reasons for having the participants randomly start with either the original or E^2GUI first. Second, most of them perceived that the E^2GUI was faster than the original. This correlates well with the results presented in Section V. Third, given that both GUIs were equally easy to learn, two-thirds of the participants indicated a preference for the E^2GUI.

The participant responses in the third section revealed user preferences for GUI design. An excerpt of statements from this section of the survey is provided in Table II. Columns 1 and 2 indicate the statement number and text, respectively. Columns 3, 4 and 5 indicate the number of participants that agreed, disagreed and were neutral toward the statement. The table indicates that input speed is important to users and that they strongly favor GUIs that let them work quicker.

One of the E^2GUI design recommendations from Section VI was to avoid vestiges of desktop systems, in particular, extraneous animations and scrollbars. On first glance at the table (No. 5, 10 and 13), it appears that users are not in favor of this change. This is not surprising since they are accustomed to desktop GUIs and expect similar features from a PDA's GUI. However, as mentioned previously, many of them are willing to make sacrifices in GUI appearance for energy efficiency (No. 3 and 16). Furthermore, although the participants indicate that they would miss the scrollbar, it is clear that the feature that matters most is the feedback that it provides (No. 10, 13 and 17). Users value the marker on the scrollbar that indicates what portion of the document the user is currently viewing. From these results, it appears that the participants are willing to give up a scrollbar for increased energy efficiency. This is especially true if they receive the same location feedback via another mechanism. Thus, higher energy efficiency, along with user satisfaction, could be achieved by making the scrollbar much thinner and using the rest of the width for page up and page down buttons.

The results from the final section of the survey verified that the participants value energy efficiency. When asked which GUI they preferred, 8 out of 12 responded that they preferred the E^2GUI. If the original GUI increased their battery lifetime by 20%, that number decreased to 3. If the E^2GUI increased their battery lifetime by 20%, 11 out of 12 responded that they would prefer the E^2GUI. Thus, it is clear that energy efficiency plays a significant role in user preferences for GUI design.

C. Summary

The results of this survey indicate that the overall user experience will be improved through the use of E^2GUI design techniques. It can be seen from Table II that there is an order of user preferences for GUI design constraints, which are listed from most important to least important below.

1) Highly productive (saves time)
2) Energy-efficient
3) Easy to learn
4) Aesthetically pleasing

The participants indicated their willingness to accept E^2GUI design techniques as long as they did not make the GUI less efficient from a performance perspective. More specifically, they are willing to make sacrifices in ease of learning as well as aesthetics for improved energy efficiency. Note that the participants' responses indicated that an E^2GUI can be as easy to learn and aesthetically pleasing as the original. Finally, we conclude from the survey that energy efficiency and user satisfaction could be increased by replacing the scrollbar with a thinner version and using the rest of the width for page up and page down buttons.

VIII. Conclusions

In conclusion, this work demonstrates that proper GUI design can improve system energy efficiency without sacrificing application performance, ease of use or aesthetics. The results indicate that E^2GUI design techniques can reduce the average system energy consumption of three benchmarks by 26.9%, 45.2% and 16.4%, respectively. Average performance is simultaneously improved by 23.7%, 34.6% and 19.3%, respectively. Thus, using E^2GUI design techniques can contribute to prolonging the battery lifetime of mobile computers. Based on these results, we proposed design techniques and made specific design recommendations for designers of mobile computers.

We believe that there is still much research that can be performed in this field. Since human perception, cognition, and motor speeds have a significant role in HCI, it would be useful to have a more thorough understanding of these processes. This is especially true for the interaction between

TABLE II
EXCERPTS FROM SURVEY SECTION 3

No.	Statement	Agree	Disagree	Neither
2	Small buttons are annoying.	9	1	2
3	I don't care what the GUI looks like if it increases my battery life by two hours.	9	3	0
4	I prefer GUIs that allow me to work quicker.	12	0	0
5	I like the hourglass, which indicates that the computer is thinking.	8	2	2
6	I would rather have a good-looking GUI than a GUI that lets me work faster.	1	10	1
10	I would miss the scrollbar because it lets me know where I am in a document.	9	2	1
13	It is easier to scroll with buttons than the scrollbar.	5	6	1
14	I would rather have an energy-efficient GUI than an easy-to-learn GUI.	6	3	3
16	It is worth extra effort to learn a GUI that will make my battery last longer.	10	1	1
17	I like knowing where I am while scrolling through a document.	9	0	3
18	Input speed matters to me.	12	0	0
19	I would rather have an easy-to-learn GUI than a good-looking GUI.	8	2	2

these processes and GUI components. In particular, we propose the following questions:

- How does the presence of active widgets (i.e., push-buttons, scrollbars, dialog boxes, etc.) affect user eye movement patterns?
- How does button size affect the duration that the user's stylus lingers at a particular point?
- Can mobile computer software be designed to mask the above effect?

Although much progress has been made in improving mobile computer energy efficiency, it is clear that there are still many opportunities for further optimization. Since many of these systems are interactive and human reaction time is significantly slower than current processor speeds, E^2GUI design provides a promising method for achieving higher energy efficiency. This work provides an overview of the problem along with description of several optimization techniques. The resulting improvements in performance and energy efficiency indicate that the use of E^2GUI design techniques enable developers to create better mobile computers.

ACKNOWLEDGMENTS

The authors would like to thank the people who participated in the experiments and survey. They would also like to thank the anonymous reviewers, whose comments significantly improved the paper.

REFERENCES

[1] J. M. Rabaey and M. Pedram, Eds., *Low Power Design Methodologies*. Kluwer Academic Publishers, 1995.

[2] J. R. Lorch and A. J. Smith, "Software strategies for portable computer energy management," *IEEE Personal Communications Magazine*, vol. 5, no. 3, pp. 60–73, June 1998.

[3] J. Flinn and M. Satyanarayanan, "Energy-aware adaptation for mobile applications," in *Proc. Symp. Operating Systems Principles*, Dec. 1999, pp. 48–63.

[4] L. Zhong and N. K. Jha, "Graphical user interface energy characterization for handheld computers," in *Proc. Int. Conf. Compilers, Architecture, and Synthesis for Embedded Systems*, Oct. 2003, pp. 232–242.

[5] ——, "Energy efficiency of handheld computer interfaces: Limits, characterization, and practice," in *Proc. ACM/USENIX Int. Conf. Mobile Systems, Applications, & Services*, June 2005.

[6] F. Gatti, A. Acquaviva, L. Benini, and B. Ricco, "Low power control techniques for TFT LCD displays," in *Proc. ACM Int. Conf. Compilers, Architecture, and Synthesis for Embedded Systems*, Oct. 2002, pp. 218–224.

[7] I. Choi, H. Shim, and N. Chang, "Low-power color TFT LCD display for handheld embedded systems," in *Proc. Int. Symp. Low Power Electronics & Design*, Aug. 2002, pp. 112–117.

[8] W.-C. Cheng and M. Pedram, "Power minimization in a backlit TFT-LCD display by concurrent brightness and contrast scaling," *IEEE Trans. Consumer Electronics*, vol. 50, no. 1, pp. 25–32, Feb. 2004.

[9] S. Pasricha, S. Mohapatra, M. Luthra, N. Dutt, and N. Subramanian, "Reducing backlight power consumption for streaming video applications on mobile handheld devices," in *Proc. First Workshop Embedded Systems for Real-Time Multimedia*, Oct. 2003.

[10] 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, & Services*, May 2003, pp. 245–258.

[11] T. Harter, S. Vroegindeweij, E. Geelhoed, M. Manahan, , and P. Ranganathan, "Energy-aware user interfaces: An evaluation of user acceptance," in *Proc. Conf. Human Factors in Computing Systems*, Apr. 2004, pp. 199–206.

[12] L. Bloom, R. Eardley, E. Geelhoed, M. Manahan, and P. Ranganathan, "Investigating the relationship between battery life and user acceptance of dynamic, energy-aware interfaces on handhelds," in *Proc. Int. Conf. Human Computer Interaction with Mobile Devices & Services*, Sept. 2004, pp. 13–24.

[13] L. Zhong and N. K. Jha, "Dynamic power optimization of interactive systems," in *Proc. Int. Conf. VLSI Design*, Jan. 2004, pp. 1041–1047.

[14] J. Lorch, "A complete picture of the energy consumption of a portable computer," Master's thesis, University of California at Berkeley, 1995.

[15] T. L. Cignetti, K. Komarov, and C. S. Ellis, "Energy estimation tools for the Palm," in *Proc. ACM Int. Workshop Modeling, Analysis and Simulation of Wireless & Mobile Systems*, Aug. 2000, pp. 96–103.

[16] J. Flinn, K. I. Farkas, and J. Anderson, "Power and energy characterization of the Itsy pocket computer (version 1.5)," Compaq Western Research Laboratory, Tech. Rep. Technical Note TN-56, Feb. 2000.

[17] S. K. Card, T. P. Moran, and A. Newell, *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Lawrence Erlbaum Assoc., 1983.

[18] R. P. Carver, *Reading Rate: A Review of Research and Theory*. San Diego, CA: Academic Press, Inc., 1990.

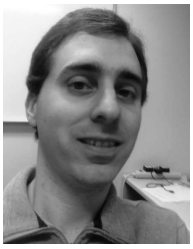
[19] R. K. Morris, K. Rayner, and A. Pollatsek, "Eye movement guidance in reading: The role of parafoveal letter and space information," *J. Experimental Psychology: Human Perception and Performance*, vol. 16, no. 2, pp. 268–281, May 1990.

[20] K. Rayner, C. M. Rotello, J. Keir, S. A. Duffy, and A. J. Stewart, "Integrating text and pictorial information: Eye movements when looking at print advertisements," *J. Experimental Psychology: Applied*, vol. 7, no. 3, pp. 219–226, Sept. 2001.

[21] P. Buser and M. Imbert, *Vision*. The MIT Press, 1992, Translated by R. H. Kay.

[22] W. E. Hick, "On the rate of gain of information," *Quarterly J. Experimental Psychology*, no. 4, pp. 11–36, 1952.

[23] R. Hyman, "Stimulus information as a determinant of reaction time," *J. Experimental Psychology*, no. 45, pp. 188–196, 1953.
[24] A. Sears and B. Schneiderman, "Split menus: Effectively using selection frequency to organize menus," *ACM Trans. Computer-Human Interaction*, vol. 1, no. 1, pp. 27–51, Mar. 1994.
[25] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement," *J. Experimental Psychology*, vol. 47, no. 6, pp. 381–391, June 1954.
[26] S. Zhai, S. Conversy, M. Beaudouin-Lafon, and Y. Guiard, "Human online response to target expansion," in *Proc. Conf. Human Factors in Computing Systems*, Apr. 2003, pp. 177–184.
[27] B. B. Bederson, "Fisheye menus," in *Proc. Symp. User Interface Software & Technology*, Nov. 2000, pp. 217–225.
[28] J. Accot and S. Zhai, "More than dotting the i's — foundations for crossing-based interfaces," in *Proc. Conf. Human Factors in Computing Systems*, Apr. 2002, pp. 73–80.
[29] I. S. MacKenzie and S. X. Zhang, "The design and evaluation of a high-performance soft keyboard," in *Proc. Conf. Human Factors in Computing Systems*, May 1999, pp. 25–31.
[30] M. T. Raghunath and C. Narayanaswami, "User interfaces for applications on a wrist watch," *Personal Ubiquitous Comput.*, vol. 6, no. 1, pp. 17–30, 2002.
[31] Sharp Zaurus SL-5500 PDA, *http://www.myzaurus.com/*.
[32] Qtopia, *http://www.trolltech.com/products/qtopia/*.



Keith S. Vallerio (S'02-M'05) received his B.S. in Computer Engineering Engineering from Villanova University, Villanova, PA, in 1999. He received his M.A. and Ph.D. degrees from Princeton University, Princeton, NJ in 2001 and 2004, respectively. His research focus has been on hardware-software co-synthesis and energy-efficient software design. He is currently a Senior Software Engineer at ALK Technologies, Inc. specializing in commercial software development for embedded Linux platforms.

Lin Zhong (S'02) received his B.S. and M.S. degrees in Electronic Engineering from Tsinghua University, Beijing, China, in 1998 and 2000, respectively. He is currently pursuing the Ph.D. degree at Princeton University, Princeton, NJ. He worked as a research intern at NEC Labs, America and Microsoft Research, Redmond in the summers of 2003 and 2004, respectively. He received the AT&T Asian-Pacific Leadership Award 2001 and the Harold W. Dodds Princeton University Honorific Fellowship for 2004-2005. For his dissertation, he works on power characterization and management of mobile systems. His other research interests include power analysis and optimization of IC & system and architecture & CAD for computing based on emerging nanometer devices.

Niraj K. Jha (S'85-M'85-SM'93-F'98) received his B.Tech. degree in Electronics and Electrical Communication Engineering from Indian Institute of Technology, Kharagpur, India in 1981, M.S. degree in Electrical Engineering from S.U.N.Y. at Stony Brook, NY in 1982, and Ph.D. degree in Electrical Engineering from University of Illinois, Urbana, IL in 1985.

He is a Professor of Electrical Engineering at Princeton University. He is an ACM Fellow. He has served as an Associate Editor of IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing. He is currently serving as an Editor of IEEE Transactions on Computer-Aided Design, IEEE Transactions on VLSI Systems, Journal of Electronic Testing: Theory and Applications (JETTA), and Journal of Embedded Computing. He has served as the guest editor for the JETTA special issue on high-level test synthesis. He has also served as the Program Chairman of the 1992 Workshop on Fault-Tolerant Parallel and Distributed Systems and the 2004 International Conference on Embedded and Ubiquitous Computing. He served as the Director of the Center for Embedded System-on-a-chip Design funded by New Jersey Commission on Science and Technology. He is the recipient of the AT&T Foundation Award and NEC Preceptorship Award for research excellence, NCR Award for teaching excellence, and Princeton University Graduate Mentoring Award. He has co-authored three books titled *Testing and Reliable Design of CMOS Circuits* (Kluwer, 1990), *High-Level Power Analysis and Optimization* (Kluwer, 1998), and *Testing of Digital Systems* (Cambridge University Press, 2003). He has also authored three book chapters. He has authored or co-authored more than 250 technical papers. He has co-authored six papers which have won the Best Paper Award at ICCD'93, FTCS'97, ICVLSID'98, DAC'99, PDCS'02, and ICVLSID'03. Another paper of his was selected for "The Best of ICCAD: A collection of the best IEEE International Conference on Computer-Aided Design papers of the past 20 years." He has received 11 U.S. patents. His research interests include low power hardware and software design, computer-aided design of integrated circuits and systems, digital system testing and distributed computing.