Benchmarking Handheld Graphical User Interface: Smoothness Quality of Experience

Ying-Dar Lin[a], Edward T.-H. Chu[b,*], Chien-Ling Wen[a], Yuan-Cheng Lai[c], I-Ching Chen[b]

[a]*Department of Computer Science, National Chiao Tung University, Taiwan*
[b]*Department of Computer Science and Information Engineering, National Yunlin University of Science and Technology, Taiwan*
[c]*Department of Information Management, National Taiwan University of Science and Technology, Taiwan*

Abstract

With the rapid growth of smartphones in the market, the smoothness of smartphones, how quickly and well smartphones react to a user's input, becomes a crucial factor consumers consider when making buying decisions. However, there is no benchmark for comparing the smoothness of one phone against another. In this paper, a handheld smoothness evaluation over regression (HSER) model was developed to make a fair evaluation. A video was first made and the several key indexes were extracted to represent behavior-based smoothness quality of services (BQoS). We built up a relationship between BQoS and behavior-based smoothness quality of experience (BQoE), and converted BQoE to handheld smoothness quality of experience. Our experiment results show that maximal frame interval and number of frame intervals are the two most critical indexes that indicate smoothness. The proposed HSER model is able to fairly evaluate the smoothness of smartphones because the error rate of the HSER model is less than 9% for a single behavior.

Keywords: QoS, QoE, Mean Opinion, GUI, Android

1. Introduction

Nowadays, mobile applications become more and more pervasive in our daily life as the number of smartphones in use accelerates. Among the variety of mobile applications, the most essential ones include web browsers, instant messaging, multimedia entertainment, intelligent and adaptive agents and mobile games. The user interfaces of these mobile applications differ from that of traditional desktop applications. All these mobile applications are triggered by multi-touch gestures, such as tap, double tap, and scroll, rather than keyboard or mouse. The smoothness of touch screen response has become one of the crucial factors considered by consumers in making their buying decisions. In this work, smoothness of smartphones is how quickly and well smartphones react to a user's input. Smartphone manufactures and mobile app designers are also interested in how their products perform with respect human-computer interaction compared to others. It has therefore become opportune to develop a way to benchmark the quality of smartphones, particular in respect of smartphone-user interaction.

A simple and intuitive method to assess user experience in mobile GUI (Graphical User Interface) is

[*]Corresponding author
Email address: `edwardchu@yuntech.edu.tw` (Edward T.-H. Chu)

to refer to hardware specifications, such as CPU speed, GPU speed and memory size. However, hardware specifications cannot fully represent the smoothness of human-device interaction because software implementation can also affect system performance. A smoothness benchmark should contain a set of user interface (UI) operations and appropriate performance indices to fairly evaluate the smoothness of smart-phones. Another method of accessing user experience is to conduct a survey using questionnaires. It will, however, be costly to conduct such a survey for each and every new smartphone. Our research goal in this work is to design a set of UI operations and representative indexes to measure the smoothness of smartphones.

1.1. Indexes of Smoothness

Frame rate is the most commonly used index to measure the smoothness of a video. The higher the frame rate becomes, the better the quality of played-back video becomes. However, Tian et al. [1, 2] found that two videos with the same average frame rate can provide very different user experiences, because one may abruptly drop a large number of frames while another may maintain an uniform frame rate. Some researchers adopted packet loss rate and network delay to evaluate the smoothness of an online game or of network streaming [3, 4, 5, 6]. Although these indexes can reflect user experience of human-interactive applications, they were not able to cover all aspects of smoothness of smartphones, especially when the smartphones under test were carried out in the same network environment. Hyeon-Ju et al. [7] also found that the off-the-shelf hardware benchmark applications, such as AnTuTu-Benchmark and SmartBench, are not able to evaluate the interaction between smartphones and users, because both hardware specifications and software can affect system performance. Traditional hardware performance metrics can thus not fully evaluate the smoothness of smartphones. As a result, it has become necessary to develop a new method to measure the smoothness of smartphones.

1.2. Handheld Smoothness Evaluation over Regression

In this work, behavior-based smoothness quality of experience (BQoE) was used to quantify the smoothness of an application. For example, making a phone call is a particular pattern of behavior, which includes a sequence of operations, such as browsing a list of contacts and tapping phone numbers. In order to measure BQoE, we first measured behavior-based smoothness quality of service (BQoS), which is service performance used to determine user satisfaction. In order to determine BQoS, a video was recorded and several key indexes were extracted. These key indexes included the mean of frame intervals (MFI), variance of frame intervals (VFI), maximal frame interval (MaxFI), frame no response (FNR) and times of maximal frame interval (TMaxFI). Since the indexes may not always be measurable, especially when the changes between frames are fast. A tool, named Ex-DOS (extraction of device operation sequence), was developed to obtain the necessary information. The previous data extraction process was repeated to obtain the same indexes from different videos that represented different user scenarios, such as calling a contact, downloading a web page or an application. Based on the BQoS obtained, we then designed a questionnaire to determine the relationship between BQoS and BQoE. Finally, the BQoE was converted to handheld smoothness QoE (HQoE) by considering how frequently each behavior is performed in daily life.

In order to evaluate the effectiveness of the proposed method, several experiments were conducted on three different smartphones, HTC hero, Huawei U8860 and Nexus S. The applicability of our handheld smoothness evaluation over regression (HSER) model has been investigated in different user scenarios. Some user scenarios are time-critical, such as making a phone call, while others are not, such as browsing a web page. The correctness of the HSER model was validated by comparing it to our questionnaire results. The rest of this work is organized as follows. Section 2 motivates this work and reviews related work for comparison. Section 3 defines the variables we used in this work and describes our problem statement. Section 4 derives the mapping from BQoS to BQoE and illustrates its implementation. Section 5 presents an evaluation. Finally, Section 6 concludes this work and indicates future directions.

2. Background and related work

2.1. Challenges of benchmarking smoothness

As far as we know, there is no standard way to benchmark the user experience of a smartphone's smoothness. Response time and frame rate per second (FPS) are two commonly used indexes to evaluate the interaction of human with smartphones. According to Jakob Nielsen's [8] and Miller's et al. [9] investigation, 0.1 second is the minimum delay that human can sense. When the delay increases to 1 second, it makes the application feel sluggish. Further, if the delay is longer than 10 seconds, users will switch to other tasks. Similar results can be found in [10], in which 0.2 second was found to be the minimum threshold for human to perceive a delay of an application.

For playing a video, a minimum of 20 FPS is recommended. Any speed below 20 FPS will result in a noticeable delay and the user will become aware of choppiness and discrete images. However, these indexes can only reflect the smoothness of one action; they are not able to evaluate the smoothness of the whole system. Furthermore, same operations with the same response time may lead to different user experiences because the changing frames displayed on a smartphone may be different, because some videos perform smoothly at an early stage while others may perform smoothly at a later stage.

2.2. Methods of recording changing frames

In order to automatically analyze the smoothness of a smartphone, it is necessary to record the interaction between a person and a smartphone. In previous work, we developed an automated GUI testing tool to test application user interfaces and to verify their functionalities [11]. Such interaction can be captured by either an internal recorder or an external camera. An internal recorder is a software agent, such as Screencast Video Recorderr, that runs on the smartphone and captures frames from the video buffer of the smartphone. Although internal recorders are easy to install and setup, they may lack the scalability for every smartphone and generate extra overheads for the system. For example, Screencast requires many memory copies [12] to capture frames from the video buffer of a smartphone. In addition, the FPS rate of the smartphones with 4.0 and 4.1 Android can be more than 60. However, the number of FPS an internal recorder can capture is usually less than 60. As a result, some frames will not be recorded and the captured video may not fully represent the original behavior of a smartphone. By contrast, the FPS of a video captured by an external camera can be more than 60, depending on the specifications of the camera. However, the quality of the captured video is sensitive to the environment, such as light intensity. More image pre-processing is also required before the captured video can be used to analyze the smoothness of a smartphone. In order to achieve a high frame rate and accurately extract the changing frames, an external recorder was utilized. Our method can also be applied to different smartphones.

2.3. Indexes of smoothness

Several indexes have been proposed to evaluate the performance of a network. For network quality, Rohani Bakar et al. [3] adopted jitter and latency to evaluate the Quality of Service (QoS). Their experiment results were validated by comparing them with the standard quality management scale defined by ITU-T. Chang et al. [4] quantified the requirements of network quality, such as network delay, packet loss rate and delay jitter, for different kinds of games. Based on network delay, delay jitter, client packet loss rate, and server packet loss rate, Chen et al. [5] developed a model to predict when players would leave a game. Chen et al. [6] also established the relationship between call duration and network quality, such as network delay, packet loss rate and delay jitter, to quantify the user satisfaction of VoIP applications. All the above-mentioned network-based indexes are not able to fully evaluate the smoothness of smartphones, because those indexes are closely related to the quality of a network. It is hard to quantify the relationship between users' interaction, such as the clicking, long pressing, and the network-based indexes.

In order to evaluate system-wide performance, several benchmarks have been developed to evaluate the performance of each hardware component of a smartphone, such as AnTuTu-Benchmark, which includes "Memory Performance", "CPU Integer Performance", "CPU Floating point Performance", "2D 3D Graphics Performance", "SD card reading/writing speed", and "Database IO Performance". Hyeon-Ju et al. [7] noted that hardware performance may not be able to fully represent software performance. Using two different strategies to implement the same software function on a platform will result in differing performances. They consequently adopted an Android utility, named Dalvik Debug Monitor Server (DDMS), to measure execution time. Although their method could evaluate the software performance, it required the source codes of the application under test. Our method, by contrast, does not need source codes and can perform black-box testing. A. Rahmati et al. [13] conducted three user studies in order to understand human-battery interaction and discover the problems in existing designs that prevent users from effectively dealing with limited battery lifetime. A. Fleury et al. [14] suggested that familiarity, convenience and annoyance are strong predictors of the preference level of methods for transferring video content from a mobile to a fixed device. However, these indexes cannot be used to evaluate the smoothness of a smartphone directly. In order to evaluate the performance of GUI and human-computer interaction, most existing proposals adopted synthetic workloads, such as floating point calculation, random string sorting, memory read/write and image rendering to assess the relative performance [15]. However, these synthetic workloads cannot fully represent the amount of work needed to be done when a user interacts with a smartphone, such as the communication among touch screen drivers, operating system and applications. Therefore, the results of existing synthetic-workload-based benchmarks are not able to objectively assess the smoothness of human-browser interaction.

Some researchers have adopted frames per second (FPS) to evaluate smoothness of smartphones [16, 17], and claimed that smartphones with higher FPS are smoother than those with lower FPS. However, for a mobile APP, the response time of user input includes the execution time of the application, the latency caused by OS kernel, interrupt handler and concurrently running tasks. FPS only represents the ability of a smartphone to process separate images per second. Furthermore, if FPS exceeds 40, it makes no difference to human beings. As a result, high FPS is not necessary for representing short response time of human-browser interaction. Tian et al. [1, 2] demonstrated that the average frame rate cannot fully reflect the smoothness of a video because the burst drop frame rate, which is the rate of suddenly dropping frames, can significantly affect user satisfaction. As a result, they extracted motion vectors (MVs) from a video to evaluate its smoothness. However, the motion vector is not suitable for the case of static frames with the external camera. For example, if dark frames on smartphones are static, the MVs of these frames should be zero. However, because of the configuration measurement environment, such as light intensity, using an external camera may result in inaccurate MVs. Xiao Feng [18] discovered that maximum frame time, frame time variance, frame rate, and frame drop rate may influence the smoothness of user interactions. He tested the same touch event of fling on two different smartphones and found that, in user experience, the smartphone with lower hardware specification performed better than that with higher hardware specification. The reason was that the frame time variance and the maximal frame time of low-end smartphones are quite a lot lower than that of high-end smart phones. Users feel sluggish when frames do not display smoothly. However, the fling operation for benchmarking can't represent every aspect of smartphone smoothness. On the contrary, in this work we extended the four indexes and translated the frame time to frame intervals for consistency. Since the frame drop rate of one operation sequence is unknown, the number of frame interval will be reduced if the frame drop rate increases. Therefore, the four indexes we used are the mean of frame intervals (MFI), variance of frame intervals (VFI), maximal frame interval (MaxFI) and number of frame intervals (NFI). Furthermore, the touch screen of smartphone is not sensitive and users will end the tasks if the delay is longer than 10 seconds. For these reasons, we also used other two indexes, frame no response (FNR) and times of maximal frame interval (TMaxFI), to evaluate the smoothness of operations. Table 1 shows the comparison of related work on indexes of smoothness.

PaperWorks [Reference #]	**Indexes**	**Drawbacks**
Video Smoothness [1] Motion Activity [2]	Average frame rates	Same frame rates may result in different user experience
VPOW-4G [3] Games QoE-Pair [4] Games QoE - Leave [5] Skypes QoE [6]	Network delay Packet loss Delay jitter	These indexes mainly focused on evaluating network performance
Our work	Mean of frame interval (MFI) Variance of frame interval (VFI) Max frame interval (MaxFI) Frame no response (FNR) Times of max frame interval (TMaxFI)	N/A

Table 1: The comparison of related work on indexes of smoothness

Paper Works [Reference #]	Classification	Objectivity
VPOW-4G [3]		
Games QoE - Leave [5]	Objective methods	Low
Skypes QoE [6]		
Games QoE-Pair [4]		Medium
Medias QoE [16]	Subjective methods	
Our work		High

Table 2: The comparison of related work on QoE models

2.4. QoE models

There are two methods of building a QoE (Quality of Experience) model: subjective and objective methods. A subjective method requires a user's opinion to assess the QoE, while an objective method adopts QoS parameters to assess the QoE. Most objective-based methods were evaluated by user's or application's behaviors. For example, Chen et al. [5, 6] collected packet traces to analyze the relationship between user behaviors and user experience, such as how long it takes for a user to end a phone call or leave a game. However, low satisfaction is not the only reason that users leave a game or end a phone. As a result, their argument may not be applied to every scenario. Rohani Bakar et al. [3] evaluated Skype using an existing standard, Standard Quality Management (SQM) defined by ITU-T P.862. Although the SQM is good for a perfect network, it may not be applicable to a network environment with packet losses and propagation delays. More QoS parameters are required to evaluate Skype-like applications. Chang et al. [4, 19] used a subjective method that adopted paired comparison to access a game's or multimedia's user satisfaction. They first asked users to compare two similar samples, such as two videos or two pictures, and select the one with better quality. Based on the users selection, they then adopted the Bradley-Terry-Luce model to determinate the probability of the users choice. The higher the probability the sample has, the greater the satisfaction the user experienced. However, the comparison is not fair because a users selection may be influenced by similar samples. For example, in the case of showing continuous similar samples, users consider the second sample as non-smooth compared to the first sample. However, in the case of showing non-continuous similar samples, users consider the second sample as individually smooth. In this work, we used yes or no question for a sample to avoid possible influences of similar samples and to evaluate the smoothness of different smartphones fairly. Table 2 shows the comparisons of related work on QoE models.

Although QoS and QoE models have been used to evaluate the performance of specific targets, finding proper metrics to construct QoS and QoE models is challenging and application-dependent. To our best knowledge, most of the existing QoS and QoE models were used to evaluate the overall performance of computer network or cloud services. For example, Wei Son et al. adopted PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity Index), and VQM (Video Quality Metric) to evaluate the quality of mobile videos [20]. Asiya Khan et al. used content type, sender bitrate, block error rate and mean burst length to predict video quality over Universal Mobile Telecommunication Systems (UMTS) networks [21]. Similarly, Abdul Hameed et al. [22] used the average number of bits per pixel in intra frames, average number of bits per pixel in inter frames and so on to evaluate video quality. All these metrics, however,

Notations	**Definitions**
B	The number of behaviors.
G	The number of the degree of smoothness.
H	The number of human operation of time sequences.
D	The number of device operation of time sequences.
a_i	An application on smartphone.
b_i	A behavior of a_i.
v_i^k	A video of b_i.
HOS_i	A human operation sequences of b_i.
DOS_i	A device operations sequences of b_i.
$HOTS = \{HOTS_i^k\ ,1 \le i \le B, 1 \le k \le G\}$ $HOTS_i^k = \left\{HOTS_{i,h}^k, 1 \le h \le H\right\}$	A set of human operation of time sequences.
$DOTS = \{DOTS_i^k\ ,1 \le i \le B, 1 \le k \le G\}$ $DOTS_i^k = \left\{DOTS_{i,d}^k, 1 \le d \le D\right\}$	A set of device operation of time sequences.
$FI_i^k = \left\{FI_{i,q}^k\ ,1 \le q \le F_I\right\}$ $FI_i^k = \left\{FI_{i,q}^k\ ,1 \le q \le F_I\right\}$	A set of frame intervals between $HOTS$ and $DOTS$
$BQoS_i^k = \{BQoS_{i,j}\ ,1 \le i \le B, 1 \le j \le P_I\ \}$ $BQoS_{i,j} = \left\{BQoS_{i,j}^k, 1 \le k \le G\right\}$	A set of performance indexes for behavior-based smoothness QoSs.
$BQoE_i^k = \left\{BQoE_{i,r}^k, 1 \le i \le B\right\}$	A set of opinion score for behavior-based smoothness QoEs.
$HQoE_i, 1 \le i \le B$	The handheld smoothness QoE.

Table 3: Definition of notations

cannot be used to quantify the smoothness of smartphones. In our BQoS and BQoE model, MaxFI and NFI were adopted to indicate smoothness.

3. Problem statements

3.1. The acquisition of BQoS

Let B denote the number of behaviors used for smoothness evaluation. The behavior b_i is defined as a sequence of operations for an application (APP) a_i (i=1, 2, ... , B). For example, making a phone call is a behavior, which includes a sequence of operations, such as browsing the list of contacts and tapping phone numbers. In order to represent the sequence of human operations in b_i, the human operation sequence (HOS_i) was used. The device operation sequence (DOS_i) is the responses to HOS_i. For example, the device operation sequence of making a phone call is a sequence of changing frames. Each HOS_i is associated with a human operation time sequence ($HOTS_i$), which stores the time instants of each human operation. Similarly, each DOS_i is associated with a device operation time sequence ($DOTS_i$), which stores the time instants of each device operation. In order to benchmark the smoothness of a smartphone for each b_i, all frame intervals named FI_i were first extracted from $HOTS_i$ and $DOTS_i$. Then, the translation function T_j was used to determine each $BQoS_{i,j}$, which is the j-th BQoS of b_i; that is, $BQoS_{i,j} = T_j\ (FI_i)$. Next, the relationship between $BQoS_i$ and $BQoE_i$ was obtained by the translation function R_i ; that is, $BQoE_i = R_i\ (BQoS_i)$. Let BQoE denote the set of all $BQoE_i$. Finally, BQoE is converted to HQoE by Eq.(10). Table 3 lists the definitions of the notations used in this work.

For example, let b_1 represent the behavior of making a phone call, which includes three operations. They are opening the APP, scrolling the contact list and dialing a number. Then, HOS_1 opens the APP, scrolling down the contact list, dialing a number, and HOS_1 is 0s, 0.5s, 1.2s, which records the starting time of each operation. Furthermore, in order to respond to HOS_1, DOS_1 pops up an app, displaying the contact list, pops up a dialog of the communication state. Each response in DOS_1 is mapped to several video frames. The timing of these video frames is recorded in DOS_1. Assuming that DOS_1 is 0.1s, 0.2s, 0.3s, 0.6s, 0.7s, 0.8s, 1.3s, 1.4s, the screen starts to change at 0.1s after the user opens the APP. The timing 0.2s and 0.3s represent the process of displaying the APP. The process of opening the APP is finally complete in 0.3s. After the user has scrolled down the contact list, the smartphone has made a series of corresponding responses to the request at 0.6s, 0.7s and 0.8s. The process of scrolling down the contact list was completed at 0.8. At the same time the smartphone starts to display the communication state at 1.3s which is complete at 1.4s. In section 4.2, the method of calculating FI_1 will be introduced.

3.2. Problem description

Let T_j denote the translation function of FI_i, which is a set of frame intervals of human operation b_i. The output of the translation function T_j is $BQoS_{i,j}$, which is the j-th BQoS of b_i. In other words, $BQoS_{i,j} = T_j\ (FI_i)$. The R_i is the translation function of $BQoS_i$. We used R_i to build up the relationship between $BQoS_i$ and $BQoE_i$. Hence, $BQoE_i = R_i\ (BQoS_i)$. The W function is used to convert BQoE into HQoE; that is, HQoE = W(BQoE). HOTS denotes the set of all $HOTS_i$ and DOTS the set of all $DOTS_i$. Given HOTS and DOTS, we aim to design functions T_j, R_i and W so that the HQoE can be determined.

4. Handheld smoothness evaluation over regression

To accurately measure every $BQoS_i$ and to build up a relationship between a $BQoS_i$ and its associated $BQoE_i$ are two key steps to determine Handheld Smoothness QoE (HQoE). This section first gives an overview of HSER, and then describes the methods used to determine each $BQoS_i$. Finally, how to build up a relationship between a $BQoS_i$ and its associated $BQoE_i$ is explained.

4.1. Overview of HSER

Figure 1 shows the overview of our approach. In order to benchmark the smoothness of a smartphone, B commonly-used behaviors were used for evaluation. For each behavior b_i, its associated $HOTS_i$ and $DOTS_i$ are was first recorded under G different CPU utilization. Let $HOTS_i^k$ denote the human operation time sequence of behavior b_i under the k-th CPU utilization. In other words, $HOTS_i$ is the set of $\{HOTS_i^1, HOTS_i^2, HOTS_i^3, ..., HOTS_i^G\}$. Similarly, $DOTS_i^k$ is the device operation time sequence of behavior b_i under the k-th CPU utilization and $DOTS_i$ is the set of $\{DOTS_i^1, DOTS_i^2, DOTS_i^3, ..., DOTS_i^G\}$. For each CPU utilization, all frame intervals were extracted. They were named FI_i^k, from $DOTS_i^k$ and

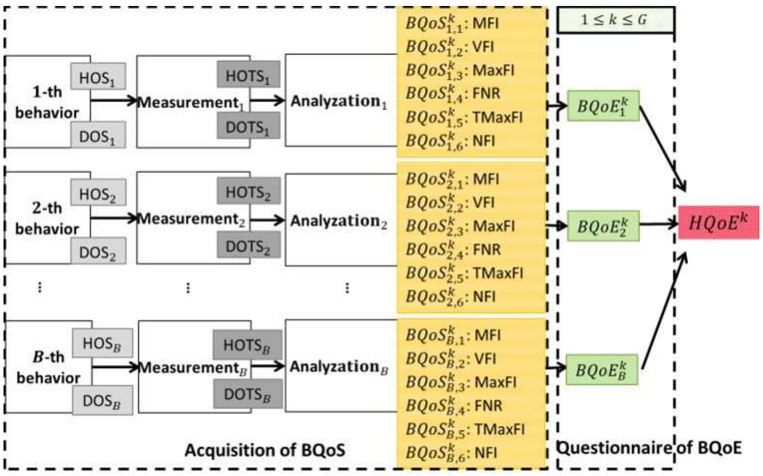


Figure 1: Flowchart of HSER

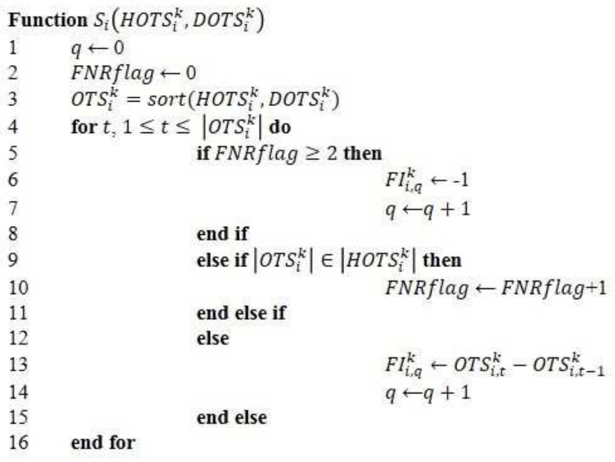
$HOTS^k_i$. The translation function T_j was then used to determine each $BQoS^k_{i,j}$, which is the j-th BQoS of b_i, under the k-th CPU utilization. In other words, $BQoS^k_{i,j} = T_j\ (FI^k_i)$. In this work, six BQoS indexes were considered. They are the mean of frame intervals (MFI), variance of frame intervals (VFI), maximal frame interval (MaxFI), frame no response (FNR), times of maximal frame interval (TMaxFI) and the number of frame intervals (NFI). A questionnaire was designed to find the relationship R_i between $BQoS_i$ and $BQoE_i$. Finally, BQoE was converted to HQoE by W function (i.e., Eq.(10)).

4.2. The acquisition of BQoS

There are two steps to obtain $BQoS^k_{i,j}$. The first step is to extract all FI^k_i, from $DOTS^k_i$and $HOTS^k_i$ and the second step is to calculate $BQoS^k_{i,j}$ by a translation function T_j.

Step1: Extract of all FI

Eight commonly-used behaviors $\{b_1, b_2, \ldots, b_8\}$ were adopted for evaluation. They were browsing web pages, viewing gallery, texting messages, listening to music, making a phone call, viewing a map, playing a game and switching between different desktops. For each behavior b_i, a keylogger tool was used to record user behavior so that $HOTS^k_i$ could be obtained under the condition of the k-th CPU utilization. In the replay stage, the user behavior was replayed and an external camera was used to capture the device responses. The captured video was then processed by our tool, Ex-DOS (see section 4.4) in order to obtain $DOTS^k_i$ under the conditions of the k-th CPU utilization. Based on $HOTS^k_i$ and $DOTS^k_i$, FI^k_i was extracted by the algorithm shown in Figure 2. Let OTS^k_i denote the time sequence which is obtained by sorting $HOTS^k_i$ and $DOTS^k_i$(line 3), $OTS^k_{i,t}$represent the t-th time instant in OTS^k_i and $FI^k_{i,q}$ notate the q-th frame in FI^k_i. There are three different cases in setting the value of each $FI^k_{i,q}$. The first case is no response; that is, there is no time instant of $DOTS^k_i$ between the current and the following time instant of $HOTS^k_i$. In this case, $FI^k_{i,q}$ is set to -1 (line 5 to 8). The second case is that $FI^k_{i,q}$ does not include the waiting time from the last operation finished to the next operation started (line 9 to 11). The third case is that $FI^k_{i,q}$ represents the response time of the operation in FI^k_i and the changing frame (line 12 to 15). For example, as shown in Figure 3(a) and Figure 3(b), let $HOTS^k_i$ be 0, 0.02, 0.04 and $DOTS^k_i$ be 0.03, 0.035, 0.055, 0.065, 0.07. The corresponding operations are triggered at



```
Function $S_i(HOTS_i^k, DOTS_i^k)$
    $q \leftarrow 0$
    $FNRflag \leftarrow 0$
    $OTS_i^k = sort(HOTS_i^k, DOTS_i^k)$
    for $t$, $1 \le t \le |OTS_i^k|$ do
        if $FNRflag \ge 2$ then
            $FI_{i,q}^k \leftarrow -1$
            $q \leftarrow q + 1$
        end if
        else if $|OTS_i^k| \in |HOTS_i^k|$ then
            $FNRflag \leftarrow FNRflag+1$
        end else if
        else
            $FI_{i,q}^k \leftarrow OTS_{i,t}^k - OTS_{i,t-1}^k$
            $q \leftarrow q + 1$
        end else
    end for
```

Figure 2: The algorithm of computing frame intervals

time instant 0, 0.02 and 0.04 respectively. After sorting $HOTS_i^k$ and $DOTS_i^k$, we obtain OTS_i^k {0, 0.02, 0.03, 0.035, 0.04, 0.055, 0.065, 0.07}. In Figure 3(c), the number with the underline is the time instant of $HOTS_i^k$. Since 0 and 0.02 are in $HOTS_i^k$, it implies that there is no response to the first operation of FI_i^k. As a result, $FI_{i,1}^k$is set to -1. In the second round, $FI_{i,2}^k$ is set to the response time of the second operation of FI_i^k; that is $FI_{i,2}^k = 0.01$. The process stops at 0.055. Hence, $FI_{i,3}^k$is 0.005 (=0.035-0.03). Similarly, the response time of the third operation of FI_i^k is $FI_{i,4}^k$, which is calculated by 0.055-0.04. Finally, $FI_{i,4}^k$ is 0.01(=0.065-0.055) and $FI_{i,6}^k$ is 0.005 (=0.07-0.065).

Step2: Calculate all BQoS

As mentioned above, we considered six BQoS indexes. They are the mean of frame intervals (MFI), variance of frame intervals (VFI), maximal frame interval (MaxFI), frame no response (FNR), times of maximal frame interval (TMaxFI) and the number of frames (NFI). We now describe how we calculated each $BQoS_{i,j}^k$ based on FI_i^k, in which j=1, 2, ... , 6. The first BQoS index is average frame interval $BQoS_{i,1}^k$ which is obtained by

$$BQoS_{i,1}^k = T_1\left(FI_i^k\right) = Avg\left(FI_i^k\right) = \frac{\sum_{q=1}^{|FI_i^k|} Excp\left(FI_{i,q}^k\right)}{|FI_i^k|} \tag{1}$$

where i is the index of behavior b_i, k is the index of CPU utilization and $|FI_i^k|$ is the number of frames in FI_i^k. If the response time is longer than 10 sec., the function Excp($FI_{i,q}$) is 10, and Excp($FI_{i,q}$) is defined as

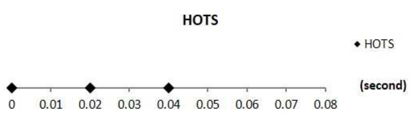
$$Excp(FI_{i,q}) = \begin{cases} FI_{i,q} & \text{,if } \ 0 \le FI_{i,q} < 10 \\ 10 & \text{,if } \ FI_{i,q} \ge 10 \ \text{ or } \ FI_{i,q} = -1. \end{cases} \tag{2}$$

The unit of $FI_{i,q}$ is a second. In order to examine how far a set of frame interval is spread out, the second BQoS index $BQoS_{i,2}^k$ is variance, which is determined by

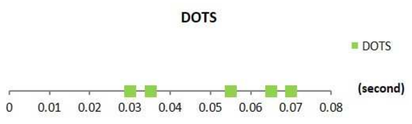
$$BQoS_{i,2}^k = T_2\left(FI_i^k\right) = AvgVar\left(FI_i^k\right) = \frac{\sum_{q=1}^{|FI_i^k|} \left(Excp\left(FI_{i,q}^k\right) - Avg\left(FI_i^k\right)\right)^2}{|FI_i^k|}. \tag{3}$$

The third BQoS index is the maximal frame interval $BQoS_{i,3}^k$, which is obtained by

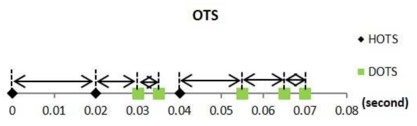
$$BQoS_{i,3}^k = T_3\left(FI_i^k\right) = Max\left(FI_i^k\right). \tag{4}$$



(a) HOTS



(b) DOTS



(c) OTS and FI

Figure 3: An example of deriving FI

Since no response can significantly affect the smoothness of a smartphone, we introduce the fourth index $BQoS^k_{i,4}$, named no response, which is defined as

$$BQoS^k_{i,4} = T_4(FI^k_i) = \sum_{q=1}^{|FI^k_i|} FNR(FI^k_{i,q}), \tag{5}$$

where FNR calculates the number of frame intervals that represent no response. Similarly, the fifth index, maximal frame interval $BQoS^k_{i,5}$, which is defined as

$$BQoS^k_{i,5} = T_5(FI^k_i) = \sum_{q=1}^{|FI^k_i|} TMaxFI(FI^k_{i,q}), \tag{6}$$

where TMaxFI calculates the number of frame intervals that are greater than 10.

Figure 4 shows two video clips with same file loading operation on two different smartphones. There are five frames in the left-hand side case (Case 1) and three frames in the right-hand side case (Case 2). Case 1 is smoother than Case 2 because more frames are displayed during the file loading process. As a result, we introduce the sixth index the number of frame intervals $BQoS^k_{i,6}$, which is defined as

$$BQoS^k_{i,6} = T_6\left(FI^k_i\right) = \left|FI^k_i\right|. \tag{7}$$

4.3. The Questionnaire for BQoE

As mentioned in sections 4.1 and 4.2, BQoS represents the service performance of a smartphone while BQoE represents the smoothness of a smartphone. Given BQoS, we aim to find the relationship between BQoS and BQoE. Because BQoE is directly related to user experience, a questionnaire is considered to be a proper tool to determine a users perception of the quality of different BQoS. A questionnaire was then used to find the relationship between BQoS and BQoE, in other words, the relationship R_i between $BQoS^k_{i,j}$ and its associated $BQoE^k_i$. In order to generate different response time for a gesture, various workloads were executed in the background, such as busy-loop. As shows in Figure 5, for each behavior, such as browsing web pages, viewing gallery or texting messages, we prepared G video clips, each of which was recorded under a specific CPU utilization. As a result, total $G \times B$ video clips were obtained. Let v^k_i denote the k-th video clip of behavior b_i. The set of video clips v^1_1, v^1_2, ... , and v^1_B

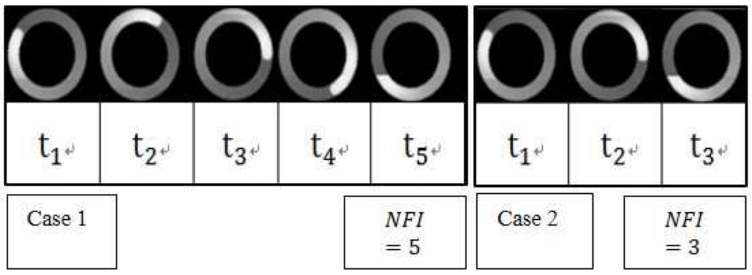


Figure 4: The factor of NFI

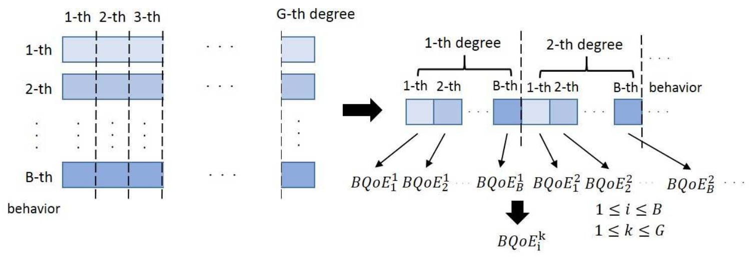


Figure 5: Idea of questionnaire

represents the response of the applications under the lightest CPU utilization. On the other hand, the set of video clips v_1^G, v_2^G, ... , and v_B^G represents the response of the applications under the heaviest CPU utilization. In our implementation, a background busy loop was adopted to generate different degrees of CPU utilization, which is used by a_i. In order to ensure the reliability of the questionnaire, more than fifty videos with different BQoS were designed to cover all possible conditions. Each volunteer was required to label the smoothness level of each video. To further ensure the reliability of the results obtained from the volunteers, three criteria were adopted to remove improper data before we applied the regression model to build the relationship between BQoS and BQoE. First, participants who did not complete the questionnaire were excluded. Second, participants who gave the same scores to all questions were excluded, and participants whose grades followed a regular pattern such as 1-2-3-1-2-3 were also dropped.

Let M denote the number of volunteers and l_r represent the r-th volunteer, in which r =1, 2, ... , M. In the first round, volunteers were asked to evaluate the smoothness of v_1^1, v_2^1, ... , and v_B^1 by answering smooth or not smooth. If l_r marks v_1^1 as smooth, then $BQoE_{1,r}^1$=1. Otherwise, $BQoE_{1,r}^1$ is set to 0. Similarly, in the second round, volunteers were asked to evaluate the smoothness of v_1^2, v_2^2, ... , and v_B^2. The same process was repeated until all $G\times B$ video clips had been evaluated by all volunteers. As a result, the corresponding scores $BQoE_i^k$ were calculated by

$$BQoE_i^k = \frac{\sum_{r=1}^{M} BQoE_{i,r}^k}{M}. \tag{8}$$

Given all $BQoS_{i,j}^k$ and $BQoE_i$, the statistic regression was used to find the relationship R between $BQoS_{i,j}^k$ (j=1, 2, ... , 6) and $BQoE_i^k$; that is

$$BQoE_i^k = R_i\left(BQoS_{i,1}^k, BQoS_{i,2}^k, BQoS_{i,3}^k, BQoS_{i,4}^k, BQoS_{i,5}^k, BQoS_{i,6}^k\right). \tag{9}$$

Finally, $HQoE^k$ is determined by a weighted function W, which is defined as

$$HQoE^k = W\left(BQoE_i^k\right) = \frac{\sum_{i=1}^{B} w_i \times BQoE_i^k}{\sum_{i=1}^{B} w_i}, \tag{10}$$

where w_i is the weight of behavior b_i.

4.4. Implementation of the Ex-DOS Tool

The purpose of tool Ex-DOS is to process a video clip v_i^k in order to obtain its associated $DOTS_i^k$. Figure 6 shows the flow of the $BQoS_{i,j}^k$ acquisition and Ex-DOS tool. For each behavior b_i, a keylogger tool was used to record user behavior FI_i^k so that we can obtain $HOTS_i^k$ under the condition of the k^{th} CPU utilization (step 1 & 2). In the replay stage, we replayed the user behavior FI_i^k and adopted an external camera with 60 fps to capture the device response DOS_i^k (step 3). The captured video v_i^k was first converted into frames by Free Video to JPG Converter Tool (DVD Video Soft [DVS], 2014) and then processed by our tool, Ex-DOS, in order to obtain $DOTS_i^k$ (step 4). Based on $HOTS_i^k$ and $DOTS_i^k$ (step 2 & 5), FI_i^k was extracted(step 6). Finally, based on all FI_i^k, each $BQoS_{i,j}^k$ was calculated (step 7).

The right-hand side of Figure 6 shows the details of step 4 that takes DOS_i^k as input and extracts $DOTS_i^k$. For each frame, a region of interest (ROI) (steps a & b) was set. We then took the current frame and the last frame for comparison (step c). Both frames were converted from color to gray frames in order to detect differences. Then, the two frames were compared pixel by pixel. If the differences of gray levels between two pixels were larger than a predefined threshold, we marked them as different pixels (step d). In order to further increase the speed of comparison, bi-level threshold recognition was used [23] (step e). Since an external camera was used to record the video, the quality of the video could have been be affected by the environment, such as light intensity. Some black pixels may be represented as gray pixels. Therefore, a medium filter was used to reduce the noises of each frame (step f). Finally, if the number of different pixels was smaller than a predefined threshold, we record the frame ID, which is the index of frames in v_i^k, and derive the time sequence $DOTS_i^k$. The same process was repeated until all frames had been processed (step g & h).

Figure 6 illustrates the three major steps of our Ex-DOS tool (numbered by 1, 2, and 3). The ROI of each frame was first set. We then processed these frames and obtain the different pixels. Finally, each $DOTS_i^k$ was obtained. In the record stage shown as Figure 6 (step 1), the keylogger tool produced a script to record FI_i^k. For each human operation in FI_i^k, the script recorded a batch of time and commands. Figure 7 shows an example of the test script. The tool automatically extracted every triggering time of human operation by detecting a specific pattern at the end of each operation (line 72, line 76 and line 80-85 of Figure 8). In the replay stage, shown as Figure 6 (step 3), computer time was used to synchronize two different time sequences for the purpose of deriving $DOTS_i^k$. One was obtained from smartphone and another was obtained from the camera. Let TS_i^k be the start time of the smartphone in b_i and TC_i^k the start time of the camera in b_i. As Figure 9 shows, a stopwatch was used to synchronize the time. Since the precision of a stopwatch is less than the time of the smartphone, the TC_i^k can be obtained by the frame rate of v_i^k in order to reduce the error. For example, TS_i^k is the 21:33:10.11 (shown

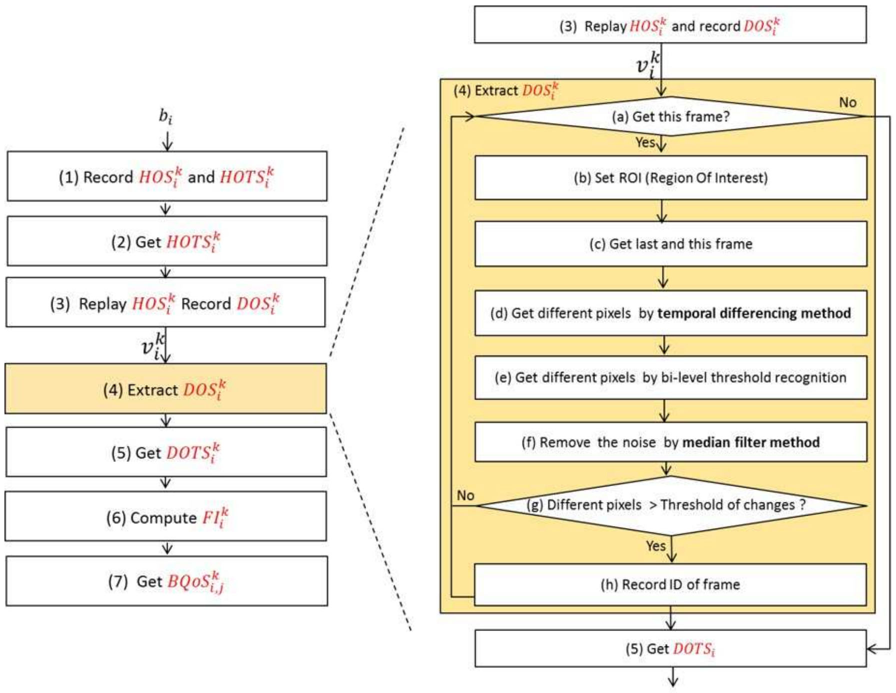
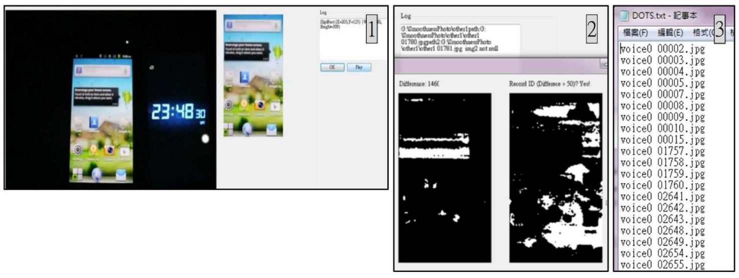

b_i
(1) Record HOS_i^k and $HOTS_i^k$
(2) Get $HOTS_i^k$
(3) Replay HOS_i^k Record DOS_i^k
v_i^k
(4) Extract DOS_i^k
(5) Get $DOTS_i^k$
(6) Compute FI_i^k
(7) Get $BQoS_{i,j}^k$
(3) Replay HOS_i^k and record DOS_i^k
v_i^k
(4) Extract DOS_i^k
(a) Get this frame?
No
Yes
(b) Set ROI (Region Of Interest)
(c) Get last and this frame
(d) Get different pixels by temporal differencing method
(e) Get different pixels by bi-level threshold recognition
(f) Remove the noise by median filter method
No
(g) Different pixels > Threshold of changes ?
Yes
(h) Record ID of frame
(5) Get $DOTS_i$

Figure 6: The flow of acquisition of BQoS and the Ex-DOS tool



```
G:\SmoothnessPhoto\other1path:G:
\SmoothnessPhoto\other1\other1
01780.jpgpath2:G:\SmoothnessPhoto
\other1\other1 01781.jpg img2 not null
```

Difference: 146(

Record ID (Difference > 50)? Yes!

```
voice0 00002.jpg
voice0 00003.jpg
voice0 00004.jpg
voice0 00005.jpg
voice0 00007.jpg
voice0 00008.jpg
voice0 00009.jpg
voice0 00010.jpg
voice0 00015.jpg
voice0 01757.jpg
voice0 01758.jpg
voice0 01759.jpg
voice0 01760.jpg
voice0 02641.jpg
voice0 02642.jpg
voice0 02643.jpg
voice0 02648.jpg
voice0 02649.jpg
voice0 02654.jpg
voice0 02655.jpg
```

Figure 7: An example of test script

as Figure 9). We obtained the time of the frame 21:33:10 in v_i^k and recorded the frame ID. Because the average time of v_i^k with 60 fps was about 0.016 second, we derived the TC_i^k is 21:33:10.112 (21:33:10+ 0.11/0.016). Based on the results, we then obtained TS_i^k and TC_i^k.

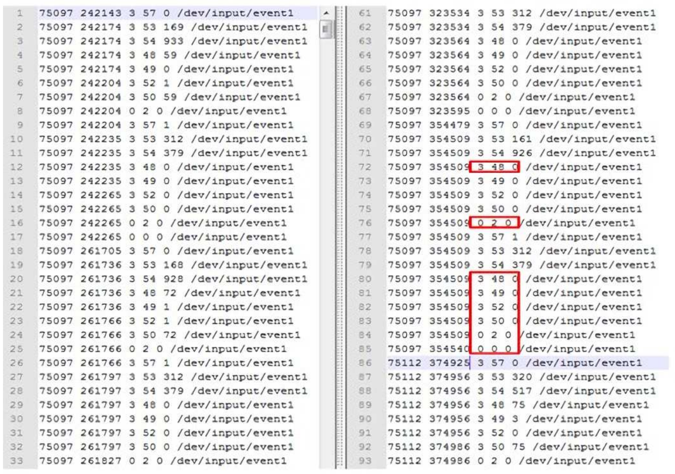
5. Evaluation

Section 5.1 sets out the experiment environment. Section 5.2 illustrates the relationship between human and device operations, and Section 5.3 investigates the correlation between BQoSs and BQoE. Section 5.4 analyzes our HSER model, and Section 5.5 evaluates the correctness of the HSER model in three different smartphones.

5.1. Testbed

5.1.1. Common user behaviors

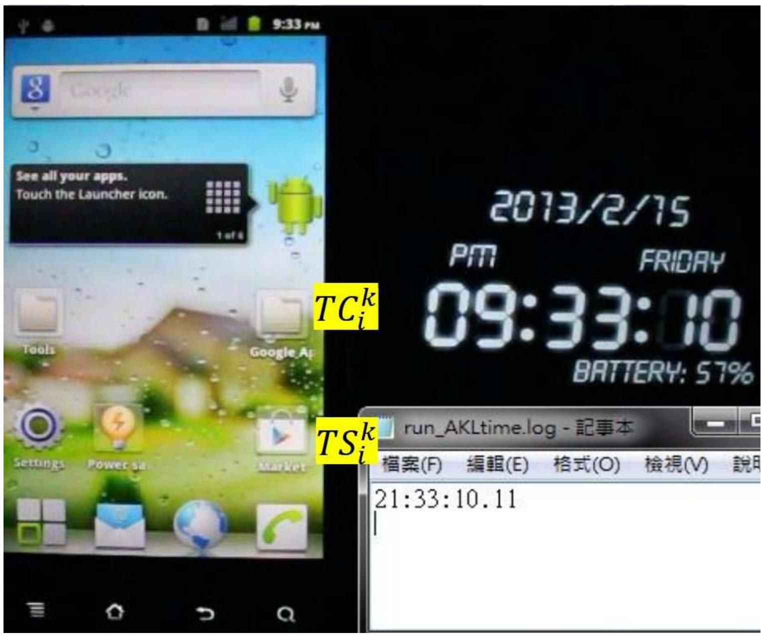
We selected eight common behaviors based on H. Verkasalo's research [24], listed in Table 4. They are: making a phone call, texting messages, browsing web pages, playing a game, viewing a map and switching



```
75097 242143 3 57 0 /dev/input/event1
75097 242174 3 53 169 /dev/input/event1
75097 242174 3 54 933 /dev/input/event1
75097 242174 3 48 59 /dev/input/event1
75097 242174 3 49 0 /dev/input/event1
75097 242204 3 52 1 /dev/input/event1
75097 242204 3 50 59 /dev/input/event1
75097 242204 0 2 0 /dev/input/event1
75097 242204 3 57 1 /dev/input/event1
75097 242235 3 53 312 /dev/input/event1
75097 242235 3 54 379 /dev/input/event1
75097 242235 3 48 0 /dev/input/event1
75097 242235 3 49 0 /dev/input/event1
75097 242265 3 52 0 /dev/input/event1
75097 242265 3 50 0 /dev/input/event1
75097 242265 0 2 0 /dev/input/event1
75097 242265 0 0 0 /dev/input/event1
75097 261705 3 57 0 /dev/input/event1
75097 261736 3 53 168 /dev/input/event1
75097 261736 3 54 928 /dev/input/event1
75097 261736 3 48 72 /dev/input/event1
75097 261736 3 49 1 /dev/input/event1
75097 261766 3 52 1 /dev/input/event1
75097 261766 3 50 72 /dev/input/event1
75097 261766 0 2 0 /dev/input/event1
75097 261766 3 57 1 /dev/input/event1
75097 261797 3 53 312 /dev/input/event1
75097 261797 3 54 379 /dev/input/event1
75097 261797 3 48 0 /dev/input/event1
75097 261797 3 49 0 /dev/input/event1
75097 261797 3 52 0 /dev/input/event1
75097 261797 3 50 0 /dev/input/event1
75097 261827 0 2 0 /dev/input/event1
```

```
75097 323534 3 53 312 /dev/input/event1
75097 323534 3 54 379 /dev/input/event1
75097 323564 3 48 0 /dev/input/event1
75097 323564 3 49 0 /dev/input/event1
75097 323564 3 52 0 /dev/input/event1
75097 323564 3 50 0 /dev/input/event1
75097 323564 0 2 0 /dev/input/event1
75097 323595 0 0 0 /dev/input/event1
75097 354479 3 57 0 /dev/input/event1
75097 354509 3 53 161 /dev/input/event1
75097 354509 3 54 926 /dev/input/event1
75097 354509 3 48 0 /dev/input/event1
75097 354509 3 49 0 /dev/input/event1
75097 354509 3 52 0 /dev/input/event1
75097 354509 3 50 0 /dev/input/event1
75097 354509 0 2 0 /dev/input/event1
75097 354509 3 57 1 /dev/input/event1
75097 354509 3 53 312 /dev/input/event1
75097 354509 3 54 379 /dev/input/event1
75097 354509 3 48 0 /dev/input/event1
75097 354509 3 49 0 /dev/input/event1
75097 354509 3 52 0 /dev/input/event1
75097 354509 3 50 0 /dev/input/event1
75097 354509 0 2 0 /dev/input/event1
75097 354540 0 0 0 /dev/input/event1
75112 374925 3 57 0 /dev/input/event1
75112 374956 3 53 320 /dev/input/event1
75112 374956 3 54 517 /dev/input/event1
75112 374956 3 48 75 /dev/input/event1
75112 374956 3 49 3 /dev/input/event1
75112 374956 3 52 0 /dev/input/event1
75112 374986 3 50 75 /dev/input/event1
75112 374986 0 2 0 /dev/input/event1
```

Figure 8: The acquisition of HOTS



```
21:33:10.11
```

Figure 9: The synchronization of time

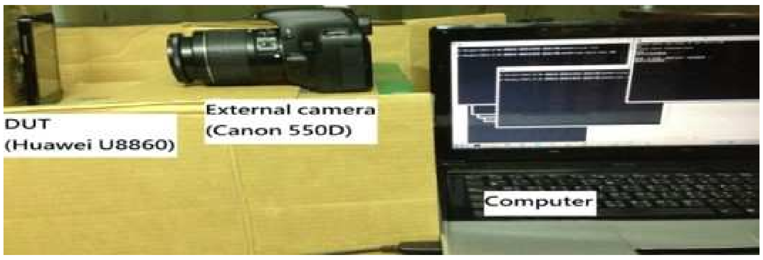


Figure 10: The experiment environment

between different desktops (i.e., "Other" in Table 4) except the multimedia type, which includes two commonly used behaviors, listing to music and viewing gallery.

5.1.2. The Experiment environment

For each behavior b_i, a keylogger tool was used to record FI_i^k of b_i so that we could obtain $HOTS_i^k$ under the condition of the k-th CPU utilization. In order to create different unsmooth scenarios, a background busy loop was used to control available CPU utilization for a_i. They are 1%, 2%, 3%, 4%, 5%, 10% and 100%; that is, G=7. The reason we chose this setting is that the CPU utilization of most operations in a smartphone require less than 10%. If available CPU utilization is greater than 10%, the application always performs smoothly in the smartphones we tested. In our experiment, CPU utilization was measured by using Linux command top, which is included in the Android Debug Bridge (ADB) toolkit [10]. An external Canon 550D camera was used to capture video of the smartphone under test. The videos were then used for further analysis. In our experiment, six widely-used Apps were used to represent real-world usage scenarios. They were Voice recorder, Messenger, Multimedia player, Browser, a 2-D animation game and Map.

In the replay stage, shown as Figure 10, we replayed the user behavior FI_i^k with a computer and Canon 550D camera to capture the device response DOS_i^k, stored in v_i^k, on device under test (DUT), a Huawei U8860. In order to eliminate the effect of environment such as light intensity, all experiments were conducted in a dark box.

Types of behavior	Percentage of using time	General operations	
Voice	34%	View the contact	
Message	21%	View the contact	
Multimedia (music and gallery)	15%	Music(7.5%): View the song lists Change the listing song Build a playlist	Gallery(7.5%): View the photos
Browser	14%	View the websites	
Games	3%	Load a game with 2D animation	
Map	3%	View the map	
Other	10%	Operate the home screen	

Table 4: The general operations for each behavior [20]

In the questionnaire stage, for the purpose of efficiency, 56 videos were posted on a website. Each video represents a behavior pertaining to specific CPU utilization. As noted above, there are 8 behaviors and 7 different CPU utilizations. The content of videos is listed in Table 4. For example, the video of

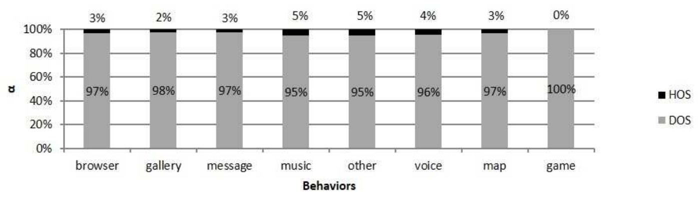


Figure 11: The value of α in different behaviors

BQoSs		Correlation r of BQoE
VFI	Linear	-0.438
	Logarithmic	-0.796
	Exponential	-0.146
	Power	0.426
MFI	Linear	-0.390
	Logarithmic	-0.723
	Exponential	-0.142
	Power	-0.721
MaxFI	Linear	-0.494
	Logarithmic	-0.705
	Exponential	-0.101
	Power	-0.705
FNR	Linear	-0.402
	Logarithmic	-0.497
	Exponential	-0.192
	Power	-0.497
TMaxFI	Linear	-0.433
	Logarithmic	-0.559
	Exponential	-0.144
	Power	-0.559
NFI	Linear	0.427
	Logarithmic	0.546
	Exponential	0.328
	Power	0.546

Table 5: The correlation r of between BQoSs and BQoE

voice behavior under 7 different CPU utilization includes the action of viewing the contact and keying the phone number. To avoid interference from another similar video, we had each volunteer grade one video at a time. The videos were cached on local disks in order to eliminate the effect of network bandwidth. We started from the first CPU utilization and asked a volunteer to grade the video. If the result was smooth, we then moved to the second CPU utilization. The process stopped when the behavior was graded as non-smooth.

5.2. Relationship between HOS and DOS

Many existing works adopted the response time of an operation to evaluate the smoothness of a smartphone. However, two operations with the same response time may lead to different user experiences because the way they change frames may be different. One may perform smoothly at an early stage while another may perform smoothly at a late stage. In order to investigate the relationship between HOS and

	VFI	MFI	MaxFI	FNR	TMaxFI	NFI
VFI	1					
MFI	0.953	1				
MaxFI	0.837	0.713	1			
FNR	0.800	0.819	0.573	1		
TMaxFI	0.792	0.757	0.746	0.804	1	
NFI	-0.701	-0.841	-0.381	-0.606	-0.478	1
Average	0.817	0.817	0.65	0.720	0.715	0.601

Table 6: The correlation r of between BQoSs

DOS, we define α_i as

$$\alpha_i = \frac{|DOS_i|}{|HOS_i| + |DOS_i|}, \tag{11}$$

in which $|HOS_i|$ is the number of operations in b_i and $|DOS_i|$ is the number of frame changes in DOS_i. As Figure 11 shows, α_i is greater than 97% in most behaviors. It implies that a behavior b_i can induce a larger number of frame changes and the response time does not reflect every aspect of smoothness of a smartphone. As a result, it is necessary to investigate the characteristics of frames when we determine the smoothness of a smartphone.

5.3. Correlation between BQoSs and BQoE

Given all $BQoS^k_{i,j}$ and $BQoE^k_i$, our goal is to find the relationship R_i between $BQoS^k_{i,j}$ (j=1, 2, ... , 6) and $BQoE^k_i$; that is

$$BQoE^k_i = R\left(BQoS^k_{i,1}, BQoS^k_{i,2}, BQoS^k_{i,3}, BQoS^k_{i,4}, BQoS^k_{i,5}, BQoS^k_{i,6}\right). \tag{12}$$

In order to reduce the complexity of R_i, the relationship between each $BQoS^k_{i,j}$ was estimated. If one BQoS index could dominate another BQoS index, the dominated BQoS index would be removed. In other words, our goal was to use as few BQoS indexes as possible to construct the function R_i. In order to estimate the relationship between each $BQoS^k_{i,j}$, we adopted coefficient of correlation r, which was determined by

$$r = \frac{\sum_{u=1}^{n}(x_u - \bar{x})(y_u - \bar{y})}{\sqrt{\sum_{u=1}^{n}(x_u - \bar{x})^2 \sum_{u=1}^{n}(y_u - \bar{y})^2}}, \tag{13}$$

where x_u is a $BQoS^k_{i,\alpha}$ sample, y_u is a $BQoS^k_{i,\beta}$ sample ($\alpha \neq \beta$) and n is the number of total samples. Table 5 shows coefficient of correlation between each BQoS and BQoE under different linear, logarithmic, exponential and power functions. As Table 5 shows, logarithmic function best fitted our data. We next investigated the logarithmic relationship among BQoSs, to see if it was possible to reduce the number of BQoS indexes. A correlation greater than 0.7 could produce the presence of collinearity and lead to a large standard error.

As Table 6 shows, VFI, MFI, FNR and TMaxFI have a strong correlation with other indexes, and the averages of the correlation of them are higher than 71.5%. In order to avoid a collinearity problem, the final BQoS indexes used to construct the relationship R_i are MaxFI ($BQoS^k_{i,3}$) and NFI ($BQoS^k_{i,6}$). In other words, users feel non-smooth if the waiting time is long and the frames are fragmented.

Maximum frame interval and frame no responses are the two most representative indexes. With our experience, a Nexus S smartphone needs more time than Huawei U8860 smartphone to process browser

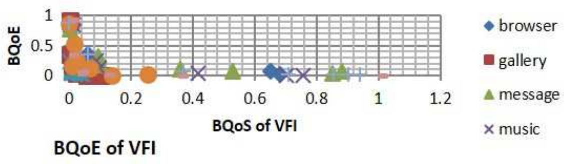
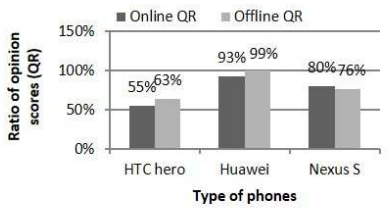
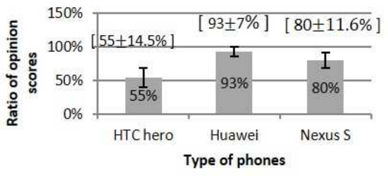


Figure 12: The relationship between the VFI and BQoE



(a) The satisfaction of smartphones



(b) The confidence interval of smartphones

Figure 13: The satisfaction and confidence interval of smartphones

request under the general CPU utilization. But, the NFI of these two smartphones is almost the same. As a result, it is desirable to include MaxFI index to further distinguish between these two smartphones. On the other hand, NFI is particularly useful if the frames are fragmented.

5.4. Analysis of the HSER model

Given all $BQoS^k_{i,3}$, $BQoS^k_{i,6}$ and $BQoE_i$, we aim to find the relationship R_i between them; that is,

$$log\left(BQoE^k_i/\left(1-BQoE^k_i\right)\right) = R_i\left(BQoS^k_{i,3},\ BQoS^k_{i,6}\right). \tag{14}$$

Multiple linear regressions were used to determine the logarithmic relationships. In order to evaluate the accuracy of the regressions result, we used the coefficient of R^2, which was obtained by

$$R^2 = \frac{\sum_{v=1}^{m}(\hat{y}_v - \bar{y})^2}{\sum_{v=1}^{m}(y_v - \bar{y})^2}, \tag{15}$$

where $\hat{y}_v$ is the predicted value of BQoE, y_v is the actual value of BQoE (the questionnaire results), and m is the total number of samples. The closer R^2 is to 1.00 the better. As shown in Table 7, R^2 is 0.528 if all behaviors were considered together. As Figure 12 shows, we further categorized behaviors into timing sensitive (such as voice and gallery behaviors), and timing non-sensitive behaviors. For example, making a phone call is a timing sensitive behavior while browsing a web page is not. According to the value of R^2, the regression performed better for timing sensitive behaviors. The correctness of regression for each individual behavior was also investigated. As shown in Table 7, the average R^2 is 0.872. In particular, for the behavior of viewing a gallery and playing a game, the R^2 increased to 0.986 and 0.973, respectively. It implies that our regression model could be used to evaluate the smoothness of a smartphone.

5.5. Evaluation of HSER model

In order to validate our HSER model for different smartphones, another round of survey was conducted. We prepared several video clips of eight operations, shown in Table 4, under normal CPU

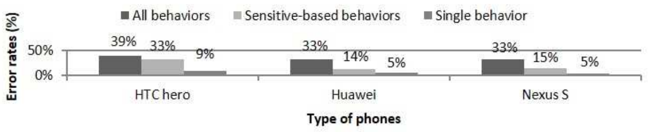


Figure 14: The error rates between the models

utilization on three very different smartphones. They are HTC hero (528 MHz CPU, 288 MB RAM, Android 2.2.1 OS), Huawei u8860 (1.4 GHz CPU, 512 MB RAM, Android 2.3 OS), and Nexus S (1 GHz, 512 MB RAM, Android 4.1.2 OS). In this survey, we had 45 volunteers to grade each video as smooth or non-smooth and used Eq.(8) and (10) to compute the results of questionnaire, denoted as QR, for each smartphone. The QRs are shown in Figure 13(a), in which Huawei U8860 is smoother than HTC hero and Nexus S. The 95% confidence interval of each survey is also shown in Figure 13(b). Considering that the online questionnaire is influenced by network delay, we collected 10 volunteers to grade each video with an offline questionnaire. As Figure 13(a) shows, the influence of the offline questionnaire results, whose ranges are in 95% confidence interval, is less than 10%. Therefore, users have good judgment even in circumstances where there is network delay. We then used our regression result, shown in Table 7, to evaluate the smoothness of each smartphone. As Figure 14 shows, the error rate, which is the error between QR and predicted result (HQoE) from our model for each smartphone, is obtained by

$$\frac{|QR - HQoE|}{|QR|}. \tag{16}$$

Our results show that for single behavior, the error rate of HSER model is less than 9%. The effectiveness of the average frame rate, which is the reciprocal of mean of frame interval (MFI) was also evaluated. The error rate of MFI is up to 71.4%. Two videos with the same MFI can provide very different user experiences, because one may abruptly drop a large number of frames while another may maintain a uniform frame rate. We argue that the HSER model is a practical solution for benchmarking the smoothness of Android smartphones.

In order to further evaluate the relationship between the smart phone speed and the effectiveness of HSER, we repeated the same set of experiments by recruiting 10 new volunteers to test two new smartphones. The volunteers' average age was 23. Most of them were students while the others were IT industry people. The two new smartphones were the LG G Pro2 (2.26GHz CPU, 3GB RAM, Android 4.4 OS) and the Samsung Galaxy Fame (1GHz CPU, 512MB RAM, Android 4.1 OS). As Table 8 shows, the average error rate of LG G Pro2 was 5.10% and that of Samsung Galaxy Fame was 10.43%. Compared with other applications, the error rate of games is bigger than others. For example, for LG G Pro2, the error rate of Game was 11% (=(0.44-0.39)/0.44), because the game flow and user interaction design can also affect user experience. Users will sense unsmoothness if the game flow is odd or strange even though the speed of the smartphone is high. By removing a game application, the average error can be further reduced to 4% to 5%. Furthermore, the error rates of voice, message and music applications are smaller than others because the user interface and interaction of these applications are relatively simple. The proposed indexes MaxFI and NFI can represent smoothness well. In addition, Figure 14 and Table 8

Type of models		R^2	avg(R^2)	Regression
All behaviors		0.528	-	$\log\left(\frac{BQoE_i}{1-BQoE_i}\right) = -1.892 - 1.24 \times \log\left(BQoS_{i,3}\right) + 0.848 \times \log\left(BQoS_{i,6}\right)$
Sensitive-based behaviors	High	0.713	0.694	$\log\left(\frac{BQoE_i}{1-BQoE_i}\right) = -2.758 - 2.01 \times \log\left(BQoS_{i,3}\right) + 1.193 \times \log\left(BQoS_{i,6}\right)$
	Low	0.675		$\log\left(\frac{BQoE_i}{1-BQoE_i}\right) = -0.578 - 2.004 \times \log\left(BQoS_{i,3}\right) + 0.765 \times \log\left(BQoS_{i,6}\right)$
Single behavior	Voice	0.813	0.872	$\log\left(\frac{BQoE_i}{1-BQoE_i}\right) = -0.823 - 2.206 \times \log\left(BQoS_{i,3}\right) + 0.548 \times \log\left(BQoS_{i,6}\right)$
	Message	0.811		$\log\left(\frac{BQoE_i}{1-BQoE_i}\right) = -3.831 - 0.969 \times \log\left(BQoS_{i,3}\right) + 1.946 \times \log\left(BQoS_{i,6}\right)$
	Gallery	0.986		$\log\left(\frac{BQoE_i}{1-BQoE_i}\right) = -24.978 - 1.189 \times \log\left(BQoS_{i,3}\right) + 9.374 \times \log\left(BQoS_{i,6}\right)$
	Music	0.951		$\log\left(\frac{BQoE_i}{1-BQoE_i}\right) = -13.563 - 0.818 \times \log\left(BQoS_{i,3}\right) + 5.655 \times \log\left(BQoS_{i,6}\right)$
	Browser	0.706		$\log\left(\frac{BQoE_i}{1-BQoE_i}\right) = -0.578 - 2.004 \times \log\left(BQoS_{i,3}\right) + 0.765 \times \log\left(BQoS_{i,6}\right)$
	Other	0.88		$\log\left(\frac{BQoE_i}{1-BQoE_i}\right) = -6.984 - 1.747 \times \log\left(BQoS_{i,3}\right) + 3.09 \times \log\left(BQoS_{i,6}\right)$
	Map	0.856		$\log\left(\frac{BQoE_i}{1-BQoE_i}\right) = -0.506 - 1.284 \times \log\left(BQoS_{i,3}\right) + 0.466 \times \log\left(BQoS_{i,6}\right)$
	Games	0.973		$\log\left(\frac{BQoE_i}{1-BQoE_i}\right) = -3.984 - 0.996 \times \log\left(BQoS_{i,3}\right) + 1.541 \times \log\left(BQoS_{i,6}\right)$

Table 7: The R Square of the models

	LG G Pro2			Samsung Galaxy Fame		
	QR	HQoE	Error Rate	QR	HQoE	Error Rate
Voice	0.56	0.57	3%	0.37	0.37	0%
Message	0.54	0.50	7%	0.40	0.39	4%
Gallery	0.58	0.55	5%	0.23	0.27	15%
Music	0.5	0.51	1%	0.37	0.38	4%
Browser	0.34	0.37	9%	0.47	0.49	5%
Other	0.54	0.53	2%	0.40	0.41	2%
Map	0.36	0.37	4%	0.17	0.18	9%
Games	0.44	0.39	11%	0.07	0.10	44%
Average	0.48	0.48	5.10%	0.31	0.32	10.41%

Table 8: Comparison of two smartphones

show that the error rate is independent of the speed of the smartphone. For example, the specifications of Nexus S and Samsung Galaxy Fame are similar, but the error rate of the Nexus S is lower than that of the Samsung Galaxy Fame. The average satisfaction of LG G Pro2 is 0.48 with a full score of 1, and that of Samsung Galaxy Fame 0.31. The confidence interval of LG G Pro2 is 11.8% and that of Samsung Galaxy Fame 10.1%. In summary, the higher the speed, the more satisfied users are with the smartphone. The confidence interval is, however, independent of the speed of smartphone.

5.5.1. Limitations of HSER model

In this work, we assigned a lower weight to the behavior of playing games when we built the HSER model, because the frequency of playing games is lower than the frequency of operating other applications [24]. Our HSER model may thus not be able to provide a reasonable scenario where a user plays games very frequently. This issue can be addressed by increasing the weight of game-playing behavior. Another limitation is that the age of the volunteers was between 18 and 25. Our HSER model may then not be representative of the perspectives of all generations. Finally, we did not conduct experiments on closed systems, such as iOS and Windows phones, because our experiments require root privilege to record the start and stop times of each user operation.

6. Conclusions and Future Work

In this work we developed a handheld smoothness evaluation over regression model to fairly benchmark the smoothness of smartphones. The applicability of this model was investigated for different user scenarios. Our experiment results showed that the correlation of mean of frame intervals, variance of frame intervals, frame no response and times of maximal frame interval was more than 71.5% in a logarithmic relationship. To avoid a collinearity problem, maximal frame interval and number of frame intervals were used as indexes for the model. Maximal frame interval and number of frame intervals were found also to be good indexes for non-smooth situations resulting from long waiting times and the fragmented frames. For individual behavior, the average R^2 was close to 1. In particular, for the behavior of viewing a gallery and playing games, R^2 increased to 0.986 and 0.973. The error rate of the proposed model was less than 9%. These findings indicate that our regression model can be used to fairly evaluate the smoothness of a smartphone. We also determined that the error rate of HTC hero smartphone (9%) was higher than two other smartphones (5%). The reasons may be variations in the way users grade the videos as either smooth or non-smooth. The same video for different users will result the different perception.

Eight different behaviors were investigated in testing the proposed model. In the future, a more comprehensive evaluation of behavioral modes will be considered. We plan to investigate other indexes and collect more users experiences in order to enhance the accuracy of our model. Possible indexes include the speed of fling and scroll operations. We also plan to improve the accuracy of the behavior capture tool Ex-DOS by detecting non-static objects in a video and to investigate the effects of networking and non-networking effects on the quality of experience. These experiments can then be extended to other platforms, such as iOS and Windows phones, to provide guidelines for modifying software to increase smoothness.

Acknowledgments

This work was supported in part by Ministry of Science and Technology (MOST) and Institute of Information Industry (III) in Taiwan.

References

[1] Tian D, Shen L, Yao Z. Motion activity based wireless video quality perceptual metric. In: Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing (ISIMP). IEEE; 2001. p. 527–530.

[2] Bertalmio M, Bertozzi AL, Sapiro G. Navier-stokes, fluid dynamics, and image and video inpainting. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). vol. 1. IEEE; 2001. p. I–355–I–362.

[3] Bakar R, Ibrahim M, Ali D. Performance measurement of VoIP over WiMAX 4G network. In: 2012 IEEE 8th International Colloquium on Signal Processing and Its Applications (CSPA). IEEE; 2012. p. 539–544.

[4] Chang YC, Chen KT, Wu CC, Ho CJ, Lei CL. Online game QoE evaluation using paired comparisons. In: 2010 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR). IEEE; 2010. p. 1–6.

[5] Chen KT, Huang P, Lei CL. Effect of network quality on player departure behavior in online games. IEEE Transactions on Parallel and Distributed Systems. 2009;20(5):593–606.

[6] Chen KT, Huang CY, Huang P, Lei CL. Quantifying Skype user satisfaction. In: Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. ACM; 2006. p. 399–410.

[7] Yoon HJ. A study on the performance of Android platform. International Journal on Computer Science and Engineering. 2012;4(4):532.

[8] Nielsen J. Usability engineering. Elsevier; 1994.

[9] Miller RB. Response time in man-computer conversational transactions. In: Proceedings of the December 9-11, 1968, fall joint computer conference, part I. AFIPS '68 (Fall, part I). ACM; 1968. p. 267–277.

[10] Android development;. Available from: `http://developer.android.com/training/articles/perf-anr.html`.

[11] Lin YD, Chu ETH, Yu SC, Lai YC. Improving the accuracy of automated GUI testing for embedded systems. IEEE Software. 2014;31(1):39–45.

[12] Wei CY, Hsieh JW, Kuo TW, Lee IH, Wu YN, Tsai MC. Resource reservation and enforcement for framebuffer-based devices. Lecture Notes in Computer Science. 2004;2968:398–408.

[13] Rahmati A, Zhong L. Human-battery interaction on mobile phones. Pervasive and Mobile Computing. 2009;5(5):465–477.

[14] Fleury A, Pedersen JS, Larsen LB. Evaluating user preferences for video transfer methods from a mobile device to a TV screen. Pervasive and Mobile Computing. 2013;9(2):228–241.

[15] Yoo W, Larson K, Baugh L, Kim S, Campbell RH. ADP: automated diagnosis of performance pathologies using hardware events. ACM SIGMETRICS Performance Evaluation Review. 2012;40(1):283–294.

[16] Clemons J, Zhu H, Savarese S, Austin T. MEVBench: A mobile computer vision benchmarking suite. In: 2011 IEEE International Symposium on Workload Characterization (IISWC). IEEE; 2011. p. 91–102.

[17] Venkata SK, Ahn I, Jeon D, Gupta A, Louie C, Garcia S, et al. SD-VBS: The San Diego vision benchmark suite. In: 2009 IEEE International Symposium on Workload Characterization (IISWC). IEEE; 2009. p. 55–64.

[18] Quantify and Optimize the User Interactions with Android Devices;. Available from: `http://software.intel.com/en-us/articles/quantify-and-optimize-the-user-interactions-with-android-devices`.

[19] Chen KT, Wu CC, Chang YC, Lei CL. A crowdsourceable QoE evaluation framework for multimedia content. In: Proceedings of the 17th ACM International Conference on Multimedia. ACM; 2009. p. 491–500.

[20] Song W, Tjondronegoro DW. Acceptability-based QoE models for mobile video. IEEE Transactions on Multimedia. 2014 April;16(3):738–750.

[21] Khan A, Sun L, Ifeachor E. QoE pediction model and its application in video quality adaptation over UMTS networks. IEEE Transactions on Multimedia. 2012 April;14(2):431–442.

[22] Hameed A, Dai R, Balas B. A decision-tree-based perceptual video quality prediction model and its application in FEC for wireless multimedia communications. IEEE Transactions on Multimedia. 2016 April;18(4):764–774.

[23] Pal NR, Pal SK. A review on image segmentation techniques. Pattern recognition. 1993;26(9):1277–1294.

[24] Verkasalo H. Analysis of smartphone user behavior. In: 2010 Ninth International Conference on Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR). IEEE; 2010. p. 258–263.

Authors' Biographies

Ying-Dar Lin is a Professor of Computer Science at National Chiao Tung University (NCTU) in Taiwan. He received his Ph.D. in Computer Science from UCLA in 1993. Since 2002. His research interests include network security, wireless communications, and embedded systems. He is an IEEE Fellow and serves on the editorial boards of several IEEE journals and magazines.

Edward T.-H. Chu is an associate professor in the Department of Electronic and Computer Science Information Engineering, National Yunlin University of Science and Technology, Taiwan. He received the Ph.D. degree in computer science in 2010 from the Department of Computer Science at National Tsing Hua University, Taiwan. His current research interests include embedded systems and real-time operating systems.

Chien-Ling Wen is a software engineer. Her research interests include embedded systems and human machine interaction. She received a MS in computer science from National Chiao Tung University, Hsinchu, Taiwan. Contact her at iolikeoi@gmail.com.

Yuan-Cheng Lai received the Ph.D. degree from the National Chiao Tung University, Hsinchu, Taiwan, in 1997. In August 2001, he joined the faculty of the Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan, where he has been a Professor since February 2008. His research interests include performance analysis, protocol design, wireless networks, and web-based applications.

I-Ching Chen is a master student in the Department of Electronic and Computer Science Information Engineering, National Yunlin University of Science and Technology, Taiwan. Her research interests include embedded systems and machine learning.