國立交通大學

網路工程研究所

碩 士 論 文

手持裝置中圖形化介面之順暢度評比

Benchmarking Handheld GUI: Smoothness QoE

研 究 生：温倩苓

指導教授：林盈達 教授

中 華 民 國 102 年 6 月

手持裝置中圖形化介面之順暢度評比

Benchmarking Handheld GUI: Smoothness QoE

研　究　生：温倩苓　　　Student: Chien-Ling Wen

指導教授：林盈達　　　Advisor: Dr. Ying-Dar Lin

國立交通大學

網路工程研究所

碩士論文

A Thesis

Submitted to Institute of Network Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science
June 2013

Hsinchu, Taiwan

中華民國 102 年 6 月

手持裝置中圖形化介面之順暢度評比

學生:温倩苓　　　　　　　　　　　　指導教授: 林盈達

國立交通大學網路工程研究所

摘要

隨著智慧型手持裝置的市占率增加,手持裝置的順暢度也逐漸成為使用者選購手機的因素之一。目前尚無一套標準的評比方法來評測智慧型手持裝置的圖形化介面之順暢度，一般常用來評估順暢度的指標為畫面更新率與反應時間，但單憑此兩個指標不足以代表使用者互動的順暢度。因此，我們分析使用者在智慧型手持裝置之使用行為的服務品質與經驗品質之關係來建立智慧型手持裝置之順暢度經驗品質模型。我們觀察的服務品質包含畫面改變的時間間隔之變異性、時間間隔之平均值、最大時間間隔、無反應次數、超過最大時間的次數、畫面改變量。經實驗證明各指標在對數關係時，除了最大時間間隔與畫面改變量，其他的指標的相關性高達 71.5%以上，易造成共線性的問題並增加模型的誤差。由此實驗結果，我們推論最長的等待時間與畫面的不連續為造成不順暢感官的原因，分別可由最大時間間隔與畫面改變量兩指標來代表。因此，最大時間間隔與畫面改變量較適合為我們服務品質的指標。另一方面，我們設計一個是非題的問卷客觀地評估經驗品質，根據問卷結果，我們可以發現使用者對於不同的使用者行為有不同程度的順暢度感官。相較於三隻手機(HTC hero、Huawei U8860 和 Nexus S)的問卷結果，我們模型的錯誤率皆低於 9%。

關鍵字: 服務品質、經驗品質、平均意見、圖形化介面、Android

Benchmarking Handheld GUI: Smoothness QoE

Student: Chien-Ling Wen **Advisor: Dr. Ying-Dar Lin**

Department of Computer Science

National Chiao Tung University

Abstract

With the rapid growth of smartphones in market, the smoothness of smartphones becomes a crucial factor considered by consumers in making their buying decisions. However, there is no standard benchmark for comparison. In addition, the commonly used smoothness indexes, such as the frame rates and response time, cannot cover all aspects of smoothness of smartphones. In order to fairly evaluate the smoothness of smartphone, we developed a handheld smoothness evaluation over regression (HSER) model to benchmark the smoothness of smartphones. We first recorded a video and extracted several key indexes to represent behavior-based smoothness quality of services (BQoS), including 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) and number of frame intervals (NFI). The correlation of MFI, VFI, FNR and TMaxFI is higher than 71.5% in logarithmic relationship. To avoid the collinearity problem which may lead to extra error, MaxFI and NFI are used to be the indexes for our HSER model. We next built up a relationship between BQoS and behavior-based smoothness quality of experience (BQoE). Finally, we converted BQoE to handheld smoothness quality of experience (HQoE). In our experiment, MaxFI and NFI are also good indexes for the "non-smooth" situations which have the long waiting time and the fragmentary frames. In addition, we tested three different smartphones, HTC hero, Huawei U8860 and Nexus S, to evaluate the applicability of HSER. Our results show that the proposed HSER model is able to fairly evaluate the smoothness of smartphones because the error rate of the HSER model is lower than 9%.

Keywords: Quality of Service, Quality of Experience, Mean Opinion, GUI, Android

致謝

非常感謝諸位師長、同學、朋友與家人在碩士論文研究生涯中不斷地給予協助與鼓勵。在這段求學的過程中，首先感謝林盈達教授、賴源正教授和朱宗賢教授的費心指導。從題目的形成至論文的產生，期間總是耐心地引導我做研究的嚴謹態度與論文寫作的技巧與知識。

其中，在做人處事方面，特別感謝林老師的教導，從進入實驗室開始，不斷訓練我們做人處事的態度以及面對問題的邏輯性思考。老師總是以身作法，包含對自己未來的規劃、時間管理的重要性等等，從老師身上學到很多對自己負責應有的態度。論文研究方面，感謝林老師與賴老師的指導，時常面臨不知道該如何做下去的窘境與看待問題的角度也不夠細膩，但每次的討論，老師們總能適時的給予方向與協助，不時也會反向鼓勵。從討論中，學習到老師們剖析問題的角度與清楚的邏輯關係，使自己剖析問題時更能全面思考。

萬分感謝朱宗賢教授不辭辛勞地為我修訂論文與撰寫方向，時常無法掌握整篇的重點、英文能力也不佳，透過老師的批改，更能體會論文精簡與論述的重要性。另外，極度感謝洪慧念教授在我的論文研究末期，耐心地指導我有關統計方面的知識，統計知識的不足是整篇論文最煎熬的一部分，很感謝老師清楚地講解與分析，獲益良多。在此非常感謝老師們的指導與教誨。

感謝交大嵌入式測試中心的甘東杰經理在期間給予我許多嵌入式系統的知識、測試項設計的精神與手持裝置設備的提供。感謝高速網路實驗室的同學、學長們，總是在我無助時伸出援手、互相討論、分享心情，很開心一起和樂的度過這兩年碩士生涯。感謝我的媽媽、弟弟、男友與大學的姊妹們，在我遭遇到挫折與苦難時，給予適當的支持和鼓勵，謝謝媽媽無條件的付出讓我能專心一致的完成學業。最後，感謝所有在問卷方面給予協助的朋友們，因為有你們，我才能順利地產生此篇論文。非常感謝大家的幫忙。

溫倩苓 謹誌於

102.06.05

Contents

List of Figures

List of Tables

Chapter 1 Introduction

Nowadays, the number of smartphones in use is accelerating. Among its variety of applications, the most essential ones include web browser, e-mail, multimedia entertainment, and mobile games. Interacting with these mobile applications differs from interacting with 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 now is one of the crucial factors considered by consumers in making their buying dec

isions. Therefore, it becomes important for both consumers and manufactures to fairly evaluate the smoothness of smartphones.

Indexes of Smoothness

Frame rate is the most commonly used index to measure the smoothness of a video. The higher the frame rate gets, 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 network streaming [3, 4, 5, 6]. Although these indexes can reflect user experience of human-interactive applications, they are not able to cover all aspects of smoothness of smartphones, especially when the smartphones under test are executed in the same network environment. Hyeon-Ju *et al*. [7] also found that the off-the-shelf hardware benchmark applications, such as AnTuTu-Benchmark [8] and SmartBench [9], are not able to evaluate the interaction between smartphones and users. This is because both hardware specifications and software can affect system performance. Traditionally hardware performance metrics cannot fully evaluate the smoothness of smartphones. As a result, it is required to develop a new method to measure the smoothness of smartphones.

Handheld Smoothness Evaluation over Regression

In this work, we adopted behavior-based smoothness quality of experience (BQoE) to quantify the smoothness of a smartphone. A behavior is defined as a sequence of operations for an application. 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 measure BQoE, we first measured behavior-based smoothness quality of service (BQoS), which is service performance used to determine user satisfaction. In order to represent BQoS, we recorded a video and then extracted several key indexes. 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, we further developed a tool, named Ex-DOS (extraction of device operation sequence), to obtain necessary information. We repeated the previous data extraction process 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 obtained BQoS, we then designed a questionnaire to determine the relationship between BQoS and BQoE. Finally, we converted the BQoE 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, we conducted several experiments on three different smartphones, HTC hero, Huawei U8860 and

Nexus S. We investigated the applicability of our handheld smoothness evaluation over regression (HSER) model 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. We validated the correctness of the HSER model by comparing it to our questionnaire results.

The rest of this work is organized as follows. Chapter 2 briefs the motivation and reviews related work to justify our problems. Chapter 3 gives the definition of variables we used in this work and describes our problem statement. Chapter 4 derives the mapping form BQoS to BQoE and illustrates our implementation. Chapter 5 presents the evaluation. Finally, chapter 6 concludes this work with future directions.

Chapter 2 Background

The chapter first describes the challenges of benchmarking smoothness, and then describes the methods used to extract the changing frames. Finally, existing works related to smoothness indexing and QoE modules are given.

2.1 Challenges of benchmarking smoothness

As far as we know, there is no standard way to benchmark the user experience of 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 *et al*. [11] and Miller's *et al*. [12] investigation, 0.1 second is the minimum delay that human can feel. 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 [13], in which 0.2 second is the minimum threshold for human to perceive a delay of an application. For playing a video, a minimum of 20 FPS is recommended. Any FPS bellows 20 will induce a noticeable delay and the user will see 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 experience because the changing frames displayed on a smartphone may be different. Dividing the changing frames into early stage and late stage. One may perform smoothly in the early stage while another may perform smoothly in the late stage.

2.2 Methods of extracting the changing frames

In order to automatically analyze the smoothness of a smartphone, it is necessary to record the interaction between human and a smartphone. This interaction can be captured by either an internal recorder or an external camera. An internal recorder is a software agent, such as Screencast Video Recorder [14], 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 induce extra overhead for the system. For example, the Screencast is not suitable for the smartphones with Nvidia's processor and requires many memory copies [15] to capture frames from the video buffer of the smartphone. In addition, the FPS of the smartphones with 4.0 and 4.1 Android platform, which is the most version on the smartphones with Android platform, can be larger than 60. However, the number of frames per second an internal recorder can capture is usually lower than 60. As a result, some frames will not be record and the captured video may not fully represent the original behavior of a smartphone. On the contrary, the FPS of a video captured by an external camera can be larger than 60, depending on the specification of camera. However, the quality of the captured video is sensitive to the environment such as light intensity. More image preprocessing is also required before the captured video can be used to analyze the smoothness of a smartphone. In this work, we adopted an external recorder in order to achieve the scalability for all smartphones and accurately extract the changing frames for avoiding losing any frames.

2.3 Related Work

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 QoS. Their experiment results were validated by comparing them with the standard quality management scale defined by ITU-T P.862. Chang *et al.* [4] quantified the requirement 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 will 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 on 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 the quality of networks. 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] mentioned 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 different performance. Hence, they adopted an Android utility, named Dalvik Debug Monitor Server (DDMS), to measure execution time. Although their method can evaluate the software performance, it requires the source codes of the application under test. Our method, on the contrary, does not need source codes and can perform black-box testing.

Tian *et al*. [1, 2] demonstrated that the average frame rate cannot fully reflect the smoothness of a video because burst drop frame rate, which is rate of the suddenly dropping frames,can significantly affect user satisfaction. As a result, they extracted motion vectors (MVs) from a video to evaluate the smoothness. However, the motion vector is not suitable for the case of static frames with the external camera. For example, some dark frames on smartphones are static. The MV can be captured more precisely by the internal recorders than the external camera. For example, the MVs of some dark frames on smartphones are zero. However, for the external camera, MVs of these frames may be mistaken because of the effect of light intensity of testing environment. Therefore, the index of MVs is not suitable for the external camera. Xiao Feng [10] discovered that the four indexes including maximal frame time, frame time variance, frame rate, and frame drop rate may influence the smoothness of user interactions. He first tested the same touch event of fling on two different smartphones. He then found that the smartphone with lower hardware specification performed better than that with higher hardware specification in user experience. The reason was that the frame time variance and the maximal frame time of the low-end smartphones are quite low. Users feel sluggish when frames do not display smoothly. However, he used only fling operation for benchmarking which can’t represent every aspect of smartphone smoothness. On the contrary, in this work, we extended the four indexes Xiao Feng found and translated the frame time to frame intervals for the consistence. However, the frame drop rate of one operation sequence is unknown. The number of frame interval will be reduced if the frame drop rate becomes higher. 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). In addition, the touch screen of smartphone is not sensitive and users will end the tasks if the delay is longer than 10 seconds. For these reason, 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 that the comparison of related work on indexes of smoothness.

Table 1 The comparison of related work on indexes of smoothness

Indexes of smoothness		
Paper Works [Reference #]	**Indexes**	**Insufficient reasons**
Video Smoothness [1]	Frame rates	Same frame rates with different users' experience
Motion Activity [2]		
VPOW-4G [3]	Network delay Packet loss Delay jitter	Hard to same network
Game's QoE-Pair [4]		
Game's QoE - Leave [5]		
Skype's QoE [6]		
This 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

QoE models

There are two kinds of methods to build up a QoE model: subjective methods and objective methods. A subjective method requires 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 the duration of time users leave a game or end a phone call. 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 application by an existing standard, Standard Quality Management (SQM) defined by ITU-T P.862. Although the SQM is good for the perfect network, it may not be applicable to a network environment with packet losses and propagation delay. More QoS parameters are required to evaluate Skype-like applications. Chang *et al*. [4, 16] used the subjective method that adopted paired comparison to access the game's or multimedia's 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 users' choice. The higher probability the sample has, the higher satisfaction user experience it is. However, the comparison is not fair because the users' selection may be influenced by similar samples. For example, in the case of showing continue similar samples, users consider the second sample as non-smooth by comparing with the first sample. However, in the case of showing non-continue similar samples, users consider the second sample as smooth individually. In this work, we used yes or no question for a sample to avoid possible influences of similar samples and fairly evaluate the smoothness of different smartphones. Table 2 shows that the comparison of related work on QoE models.

Table 2 The comparison of related work on QoE models

<table>
<tr><th colspan="4">QoE models</th></tr>
<tr><th>Paper Works [Reference #]</th><th>Quantifiable method of users' experiences</th><th colspan="2">Objectivity</th></tr>
<tr><td>VPOW-4G [3]</td><td rowspan="3">Objective methods</td><td rowspan="3" colspan="2">Low</td></tr>
<tr><td>Game's QoE - Leave [5]</td></tr>
<tr><td>Skype's QoE [6]</td></tr>
<tr><td>Game's QoE-Pair [4]</td><td rowspan="3">Subjective methods</td><td rowspan="2">Continuous similar samples</td><td rowspan="2">Medium</td></tr>
<tr><td>Media's QoE [16]</td></tr>
<tr><td>This work</td><td>Non- Continuous similar samples</td><td>High</td></tr>
</table>

Chapter 3 Problem Statements

3.1 Terminology

Let B denote the number of behaviors used for smoothness evaluation. A behavior b_i is defined as a sequence of operations for an application (APP) a_i $(i = 1,2, \ldots, B)$. 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. We use *human operation sequence (HOS_i)* to represent the sequence of human operations in b_i. In addition, 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, we first extract all frame intervals, named FI_i, from $HOTS_i$ and $DOTS_i$. We then use the translation function T_j 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, we find the relationship between $BQoS_i$ and $BQoE_i$ by the translation function R_i; that is, $BQoE_i = R_i(BQoS_i)$. Let $BQoE$ denote the set of all $BQoE_i$. We finally convert $BQoE$ to $HQoE$ by the function W; that is, $HQoE = W(BQoE)$. Table 3 lists the definition 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 up a phone. Then, HOS_1 is {opening the APP, scrolling the contact list, dialing up a phone}, and $HOTS_1$ is {0s, 0.5s, 1.2s}, which records the starting time of each operation. In addition, in order to respond to HOS_1, DOS_1 is {popping up app, displaying the contact list, popping up a dialog of communication state}. Each response in DOS_1 is mapped to several video frames. The timing of these video frames is recorded in $DOTS_1$. Assuming that $DOTS_1$ is {0.1s, 0.2s, 0.3s, 0.6s, 0.7s, 0.8s, 1.3s, 1.4s}, it implies that the screen starts to change at 0.1s after the user opening the APP. The timing 0.2s and 0.3s represent the process of showing up the APP. The process of opening the APP finally completes as 0.3s. In addition, after the user scrolled up the contact list, the smartphone made a series of corresponding responses to the request at 0.6s, 0.7s and 0.8s. The process of scrolling the contact list was completed at 0.8. Similarly, the smartphone started to display a dialog of communication state at 1.3s and completed at 1.4s. In Chapter 4.2, we will introduce the method of calculating FI_1.

3.2 Problem Description

Let $HOTS$ denote 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.

Table 3. Definition of notations

Notations	Definitions
B	The number of behaviors.
G	The number of the degree of the CPU utilization.
P_I	The number of performance indexes.
F_I	The number of frame intervals.
M	The number of volunteers.
a_i	An application of *i-th* behavior on a smartphone.
b_i,	A sequence of operations for a_i.
v_i^k	A video of b_i under the *k-th* CPU utilization.
w_i	A weight of b_i.
$HOS_i = \{HOS_{i,h}, 1 \le h \le H\}$	Human operation sequence of b_i.
$DOS_i = \{DOS_{i,d}, 1 \le d \le D\}$	Device operation sequence of b_i.
$HOTS_i = \{HOTS_i^k, 1 \le i \le B, 1 \le k \le G\}$	A set of human operation of time sequences.
$DOTS_i = \{DOTS_i^k, 1 \le i \le B, 1 \le k \le G\}$	A set of device operation of time sequences.
OTS_i^k	A set of interleaved operation of time sequences by $HOTS_i$ and $DOTS_i$.
$FI = \{FI_i^k, 1 \le i \le B, 1 \le k \le G\}$ $FI_i^k = \{FI_{i,q}^k, 1 \le q \le F_I\}$	A set of frame intervals between $HOTS$ and $DOTS$.
$BQoS_i^k = \{BQoS_{i,j}, 1 \le i \le B, 1 \le j \le P_I\}$ $BQoS_{i,j} = \{BQoS_{i,j}^k, 1 \le k \le G\}$	A set of performance indexes for behavior-based smoothness QoSs.
$BQoE_i^k = \{BQoE_{i,r}^k, 1 \le k \le G, 1 \le i \le B, 1 \le r \le M\}$	A set of opinion score for behavior-based smoothness QoEs.
$HQoE^k, 1 \le k \le G$	The handheld smoothness QoE.
T_j	A translation function for FI_i^k to $BQoS_{i,j}^k$.
R_i	A translation function for $BQoS_i^k$ to $BQoE_i^k$.
W	A translation function for $BQoE_i^k$ to $HQoE^k$.
TC_i^k	The start time of the camera in b_i.
TS_i^k	The start time of the smartphone in b_i.
QR	The statistics of questionnaire result for each smartphone.

Chapter 4 Handheld Smoothness Evaluation over Regression

Accurately measuring every $BQoS_i$ and building up a relationship between a $BQoS_i$ and its associated $BQoE_i$ are two key steps to determine Handheld Smoothness QoE ($HQoE$). In this chapter, we first give an overview of HSER. We then describe the methods used to determine each $BQoS_i$ and explain how to build up a relationship between a $BQoS_i$ and its associated $BQoE_i$.

4.1 Overview of HSER

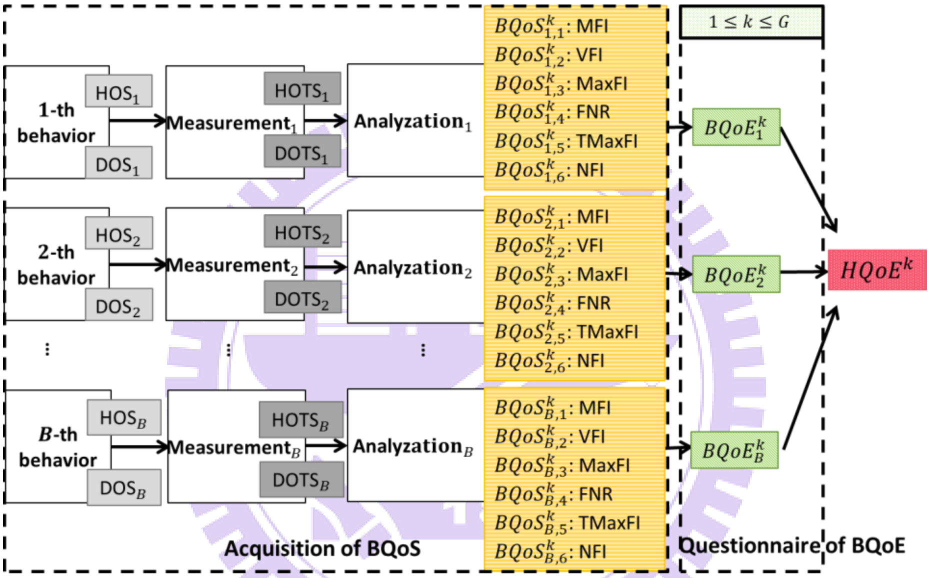


Figure 1. Flowchart of HSER

Figure 1 shows the overview of our approach. In order to benchmark the smoothness of a smartphone, we adopted B commonly-used behaviors for evaluation. For each behavior b_i, we first record its associated $HOTS_i$ and $DOTS_i$ 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 $\{HOTS_i^1, HOTS_i^2, HOTS_i^3, \ldots, HOTS_i^G\}$. 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 $\{DOTS_i^1, DOTS_i^2, DOTS_i^3, \ldots, DOTS_i^G\}$. For each CPU utilization, we extract all frame intervals, named FI_i^k, from $DOTS_i^k$ and $HOTS_i^k$. We then use the translation function T_j to determine each $BQoS_{i,j}^k$, which is the *j-th* $BQoS$ of b_i, under the *k-th* CPU utilization. In other words, we have $BQoS_{i,j}^k = T_j(FI_i^k)$. In this work, we consider 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 frame intervals (NFI). Next, we design a questionnaire to find the relationship R_i between $BQoS_i$ and $BQoE_i$. We finally convert $BQoE$ to $HQoE$ by the function W.

4.2 The Acquisition of BQoS

There are two steps to obtain $BQoS_{i,j}^k$. The first step is to extract all FI_i^k, from $DOTS_i^k$ and $HOTS_i^k$ and the second step is to calculate $BQoS_{i,j}^k$ by a translation function T_j.

Step1: Extraction of all FI

We adopted eight commonly-used behaviors $\{b_1, b_2, \ldots, b_8\}$ for evaluation. They are 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, we used Android keylogger (AKL) [17] to record user behavior so that we can obtain $HOTS_i^k$ under the condition of the *k-th* CPU utilization. In the replay stage, we replayed the user behavior and adopted an external camera to capture the device responses. The captured video is then processed by our tool, Ex-DOS (See Chapter 4.4) in order to obtain $DOTS_i^k$ under the condition of the *k-th* CPU utilization. Based on $HOTS_i^k$ and $DOTS_i^k$, we extracted FI_i^k by the algorithm shown in Figure 2.

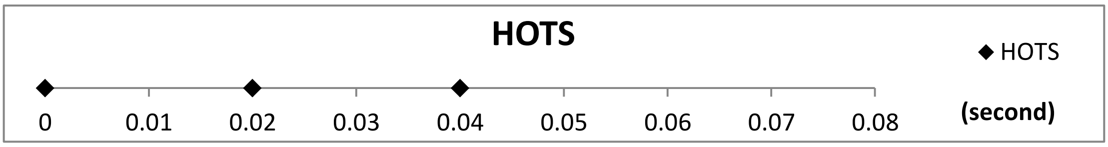
Let OTS_i^k denote the time sequence which is obtained by sorting $HOTS_i^k$ and $DOTS_i^k$ (line 3), $OTS_{i,t}^k$ represent the *t-th* time instant in OTS_i^k and $FI_{i,q}^k$ notate the *q-th* frame in FI_i^k. There are three different cases in setting the valuate of each $FI_{i,q}^k$.

The first case is no response; that is, there is no time instant of $DOTS_i^k$ between current and next time instant of $HOTS_i^k$. In this case, $FI_{i,q}^k$ is set to -1 (line 5 to 8). The second case is that $FI_{i,q}^k$ is not including the waiting time from the last operation finished to the next operation started (line 9 to 11). The third case is that $FI_{i,q}^k$ represents the response time of the operation in HOS_i^k and the changing frame (line 12 to 15).

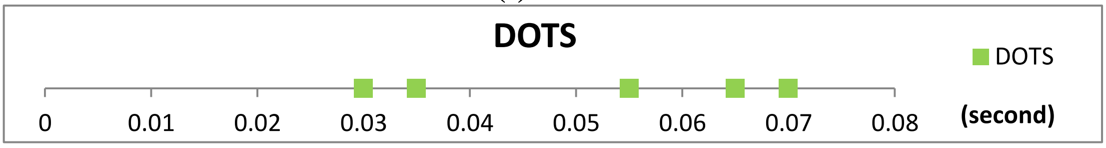
For example, as shows in Figure 3(a) and Figure 3(b), let $HOTS_i^k$ be {0, 0.02, 0.04} and $DOTS_i^k$ be {0.03, 0.035, 0.055, 0.065, 0.07}. Three triggered time of corresponding operations are at time instant 0, 0.02 and 0.04 respectively. After sorting $HOTS_i^k$ and $DOTS_i^k$, we can obtain OTS_i^k {0, 0.02, 0.03, 0.035, 0.04, 0.055, 0.065, 0.07} shown in Figure 3(c) and the number with the baseline means 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 HOS_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 the second operation of HOS_i^k; that is $FI_{i,2}^k$ = 0.01. The process stops until 0.055. Hence, $FI_{i,3}^k$ is 0.005 (=0.035-0.03). Similarly, the response time of the third operation of HOS_i^k is $FI_{i,4}^k$, which is calculated by 0.055-0.04. Finally, $FI_{i,5}^k$ is 0.01(=0.065-0.055) and $FI_{i,6}^k$ is 0.005 (=0.07-0.065).

Function $S_i(HOTS_i^k, DOTS_i^k)$
1 $q \leftarrow 0$
2 $FNRflag \leftarrow 0$
3 $OTS_i^k = sort(HOTS_i^k, DOTS_i^k)$
4 **for** $t,\ 1 \le t \le |OTS_i^k|$ **do**
5 **if** $FNRflag \ge 2$ **then**
6 $FI_{i,q}^k \leftarrow -1$
7 $q \leftarrow q + 1$
8 **end if**
9 **else if** $|OTS_i^k| \in |HOTS_i^k|$ **then**
10 $FNRflag \leftarrow FNRflag+1$
11 **end else if**
12 **else**
13 $FI_{i,q}^k \leftarrow OTS_{i,t}^k - OTS_{i,t-1}^k$
14 $q \leftarrow q + 1$
15 **end else**
16 **end for**

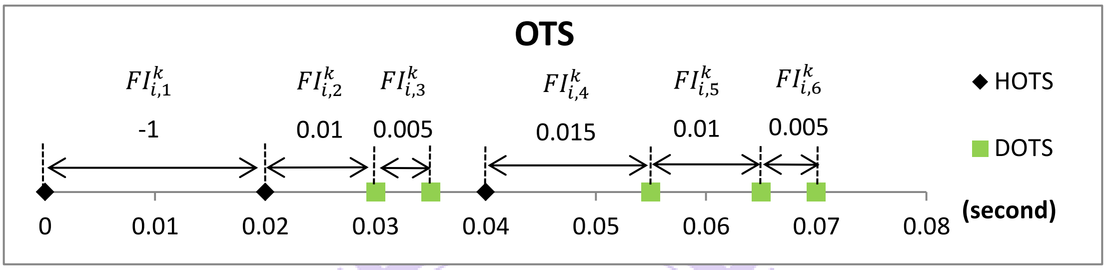
Figure 2 The algorithm of computing frame intervals



(a) HOTS



(b) DOTS



(c) OTS and FI

Figure 3 An example of deriving FI

Step2: Derivation of all $BQoS$

As mentioned before, 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). In the this section, we describe how we calculated each $BQoS_{i,j}^k$ based on FI_i^k, in which $j = 1,2,\dots,6$.

The first $BQoS$ index $BQoS_{i,1}^k$ is average frame interval which is obtained by

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

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. In addition, the function $Excp(FI_{i,q})$, which sets the time as 10 seconds in the no response case and the larger time case, is defined as

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

The unit of $FI_{i,q}$ is second. If $FI_{i,q}$ is larger than 10 seconds, we set $FI_{i,q}$ to 10. 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

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

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

$$BQoS_{i,3}^{k} = T_3(FI_i^k) = Max(FI_i^k).$$

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

$$BQoS_{i,4}^{k} = T_4(FI_i^k) = \sum_{q=1}^{|FI_i^k|} FNR(FI_{i,q}^k),$$

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

$$BQoS_{i,5}^{k} = T_5(FI_i^k) = \sum_{q=1}^{|FI_i^k|} TMaxFI(FI_{i,q}^k),$$

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

Figure 4 show two video clips of the 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. The sixth index $BQoS_{i,6}^{k}$ is the number of frame intervals, which is defined as

$$BQoS_{i,6}^{k} = T_6(FI_i^k) = |FI_i^k|.$$

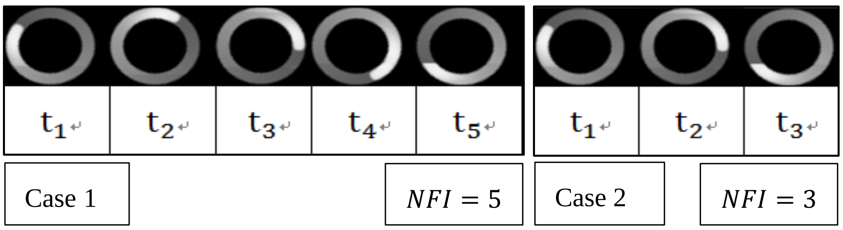


Figure 4 The factor of NFI

4.3 The Questionnaire for BQoE

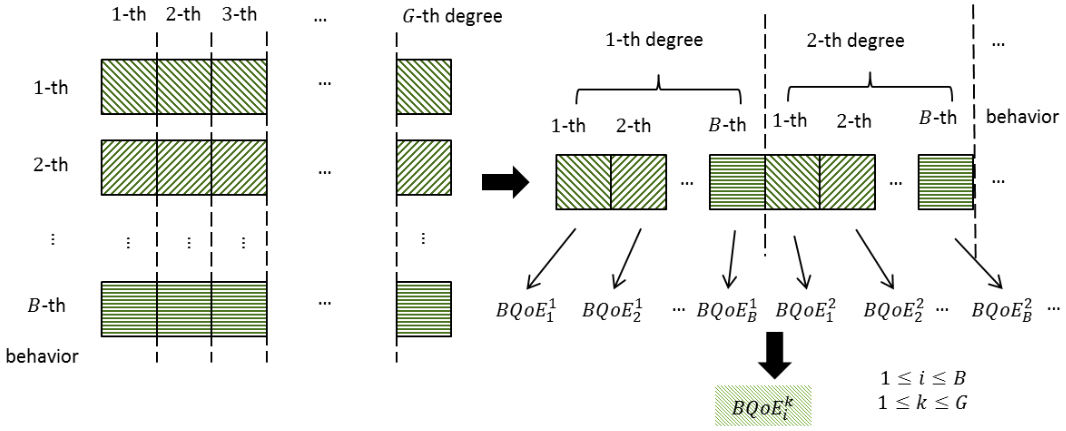


Figure 5 Idea of questionnaire

We designed a questionnaire to find the relationship R_i between $BQoS_{i,j}^k$ and its associated $BQoE_i^k$. 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 record under a specific CPU utilization. As a result, we have total $G \times B$ video clips. Let v_i^k denote the *k-th* video clip of behavior b_i. The set of video clip $\{v_1^1,\ v_2^1,\ \ldots$, and $v_B^1\}$ represent the response of the applications under the lightest CPU utilization. On the other hand, the set of video clip $\{v_1^G,\ v_2^G,\ \ldots$,and $v_B^G\}$ represent the response of the applications under the heaviest CPU utilization. In our implementation, we adopted a background busy loop to generate different CPU utilization, which is used by a_i.

Let M denote the number of volunteers and l_r represent the *r-th* volunteer, in which $r = 1,2, \ldots, M$. At the first round, we asked volunteers to evaluate the smoothness of $v_1^1,\ v_2^1,\ \ldots$, 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, we set $BQoE_{1,r}^1$= 0. Similarly, in the second, volunteers were asked to evaluate the smoothness of $v_1^2,\ v_2^2, \ldots$, and v_B^2. We repeated the same process until all $G \times B$ video clips were evaluated by all volunteers. As a result, we calculated the corresponding scores $BQoE_i^k$ by

$$BQoE_i^k = \frac{\sum_{r=1}^{M} BQoE_{i,r}^k}{M}. \quad (1)$$

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

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

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

$$HQoE^k = W\left(BQoE_i^k\right) = \frac{\sum_{i=1}^{B} w_i \times BQoE_i^k}{\sum_{i=1}^{B} w_i}, \quad (2)$$

where w_i is the weight of behavior b_i.

4.4 Implementation of the Ex-DOS Tool

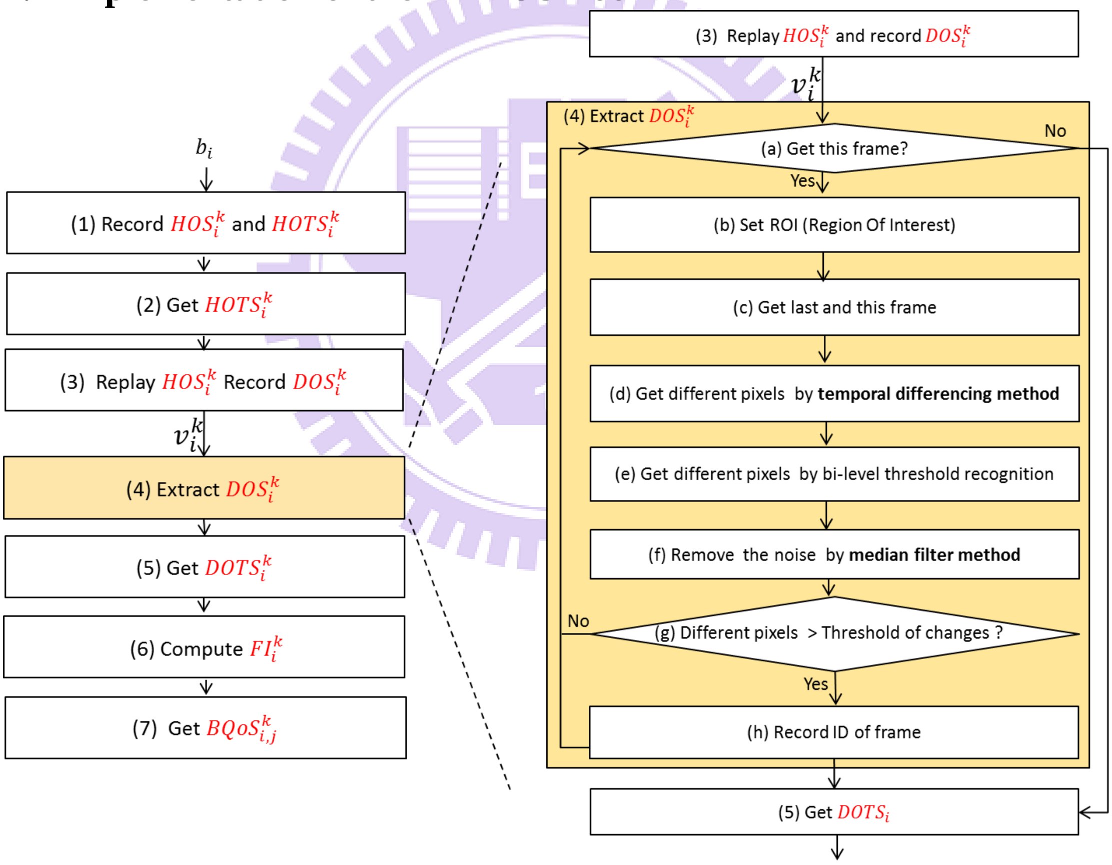


Figure 6 The flow of acquisition of BQoS and the Ex-DOS tool

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 the position of Ex-DOS tool. For each behavior b_i, we used Android keylogger (AKL) to record user behavior HOS_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 HOS_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 [18] 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), we extracted FI_i^k (step 6). Finally, based on all FI_i^k, we calculate each $BQoS_{i,j}^k$ (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, we set a region of interest (ROI) (step a & b). We then got the current frame and the last frame for comparison (step c). Both frames were converted from color frames to gray frames in order to detect the difference. We compared pixel by pixel in two frames. If the differences of gray levels between two pixels are larger than a predefined threshold, we marked them as different pixels (step d). In order to further increase the speed of comparison, we adopted bi-level threshold recognition[19] (step e). Since we adopted an external camera to record the video, the quality of the video may be affected by the environment such as light intensity. Some black pixels may be represented as gray pixels. Therefore, we further adopted a medium filter 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 $DOTS_i^k$. We repeat the same process until all frames have been processed (step g & h).

Figure 7 shows the three major steps of our Ex-DOS tool (numbered by 1, 2, and 3). We first set the ROI of each frame (step a & b in Figure 6). We then processed these frames (step d-f) and obtain the different pixels. Finally, each $DOTS_i^k$ was obtained.

In the record stage shown as Figure 6 (step 1), the AKL produced a script to record HOS_i^k. For each human operation in HOS_i^k, the script recorded a batch of time and commands. Figure 8 shows an example of test script. We 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), we use computer time to synchronize two different time sequences for the purpose of deriving $DOTS_i^k$. One is obtained from smartphone and another is obtained from the camera. Let TS_i^k as the start time of the smartphone in b_i and TC_i^k as the start time of the camera in b_i. As Figure 9 shows, we first used a stopwatch to synchronize the time. However, the precision of stopwatch is lower than the time of the smartphone. To reduce the error, we derive the TC_i^k by the frame rate of v_i^k. For example, TS_i^k is the 21:33:10.11 (shown as Figure 9). We first find the time of the frame is 21:33:10 in v_i^k and record the frame ID. Because the average time of v_i^k with 60 fps is about 0.016 second, we can derive the TC_i^k is 21:33:10.112 (21:33:10+ 0.11/0.016). We then obtained TS_i^k and TC_i^k respectively.

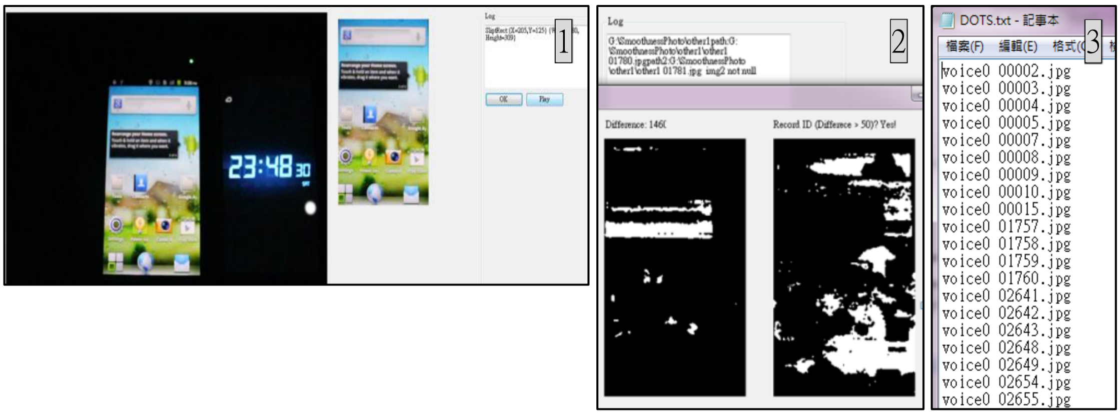
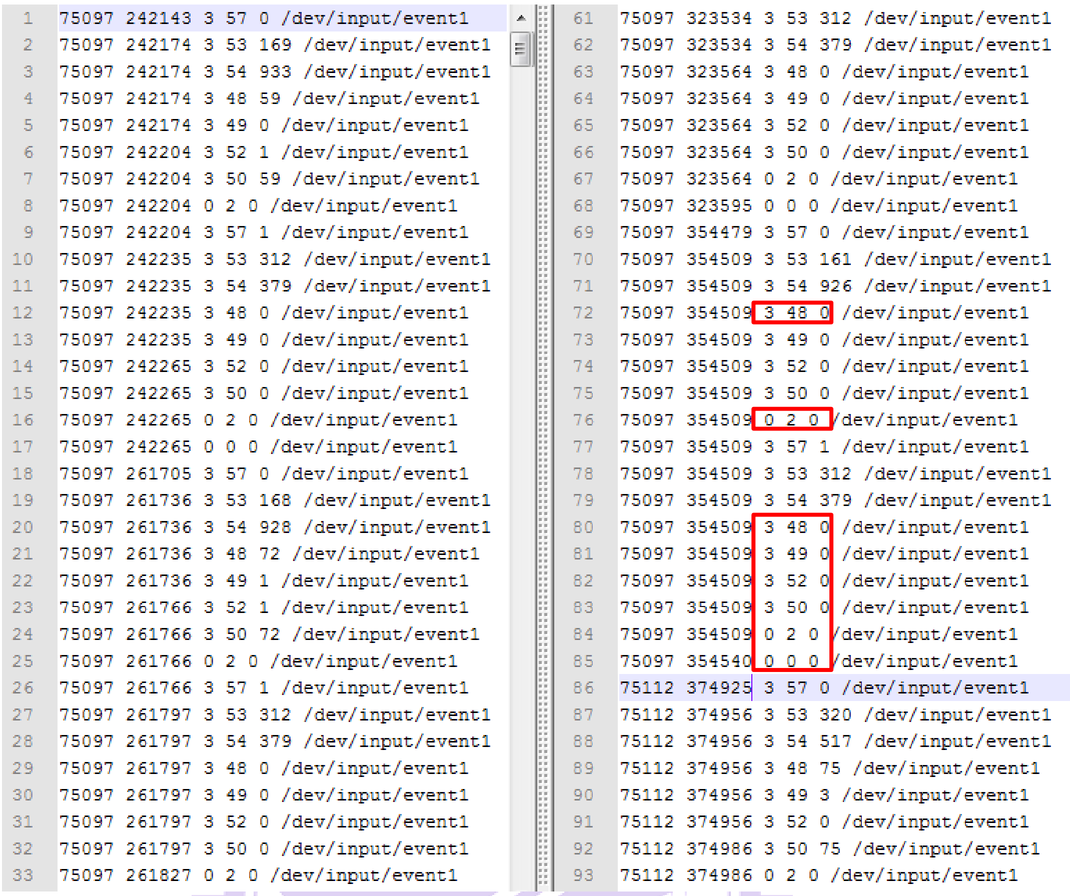


Figure 7 The flow of Ex-DOS tool



```
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

Figure 9 The synchronization of time

Chapter 5 Evaluation

In this chapter, we first introduce the experiment environment in Section 5.1. Next, Section 5.2 illustrates the relationship between human and device operations. Section 5.3 investigates the correlation between BQoSs and BQoE. Section 5.4 analyzes our HSER model. Finally, Section 5.5 evaluates the correctness of the HSER model in three different smartphones.

5.1 Testbed

Our testbed includes a host PC, a Huawei U8860 smartphones with the AKL agent and a Canon 550D camera with non-interlaced 720 lines at 60 FPS.

Common user behaviors

According the Verkasalo‘s research [20], the most used behaviors are voice (34%), message (21%), multimedia (15%), browser (14%), games (3%), map (3%) and other (10%) respectively. Table 4 shows the seven common types of behaviors. Based on the seven common types of behavior, we adopted eight behaviors in our experiment. They are making a phone call, texting messages, browsing web pages, playing a game, viewing a map and switching between different desktops (i.e. “other” in Table 4) respectively except the multimedia type, which is including two commonly used behaviors, listing to music and viewing gallery; that is $B = 8$.

The experiment environment

For each behavior b_i, we used Android keylogger (AKL) to record HOS_i^k of b_i so that we can obtain $HOTS_i^k$ under the condition of the *k-th* CPU utilization. In our implementation, we adopted a background busy loop control available CPU utilization for a_i. They are 1%, 2%, 3%, 4%, 5%, 10% and 100%; that is $G = 7$. The reason we choose this setting is that the CPU utilization of most operations in a smartphone require less than 10%. If available CPU utilization is larger than 10%, the application always performs smoothly in the smartphones we tested.

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

In the questionnaire stage, for the purpose of efficiency, we posted 56 videos on an online website designed with PHP. Each video represents a behavior under a specific CPU utilization. As the mentioned above, there are 8 behaviors and 7 kind of CPU utilization. The content of videos is shown in Table 4. For example, the videos of voice behavior under 7 kind of CPU utilization include the action of viewing the contact and keying the phone number. To avoid the interference from another similar video, a volunteer graded a video at a time. Each volunteer graded the videos which are all kind of behaviors under *1-th* CPU utilization with "smooth" or "non-smooth" first and reproduce the process that grading the videos of the behaviors which were graded with "smooth" until the videos of all behaviors are graded with "non-smooth". In addition, to avoid users confusing the network delay with the "non-smooth" situation, we announced that users acted the judgement when the widgets of the frame have moved.

Table 4 The general operations for each behavior

Types of behavior	Percentage of using time	General operations	
Voice	34%	◆ View the contact ◆ Key the phone number	
Message	21%	◆ View the contact ◆ Key the messages	
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 ◆ Search the nearby places	
Other	10%	◆ Operate the home screen	

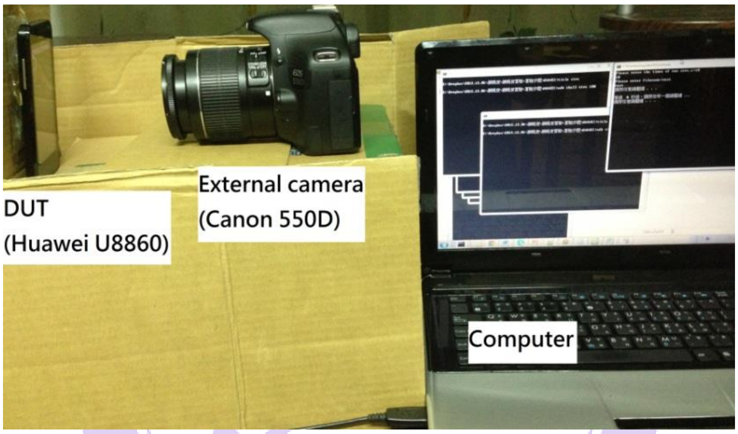


Figure 10 The experiment environment

5.2 Relationship between HOS and DOS

Many existing work 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 experience because the way they change frames may be different. One may perform smoothly in the early stage while another may perform smoothly in the late stage. In order to investigate the relationship between HOS and DOS, we defined α_i as

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

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 larger than 97% in most behaviors. It implies that a behavior b_i can induce a larger number of frame changes and the response time cannot 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.

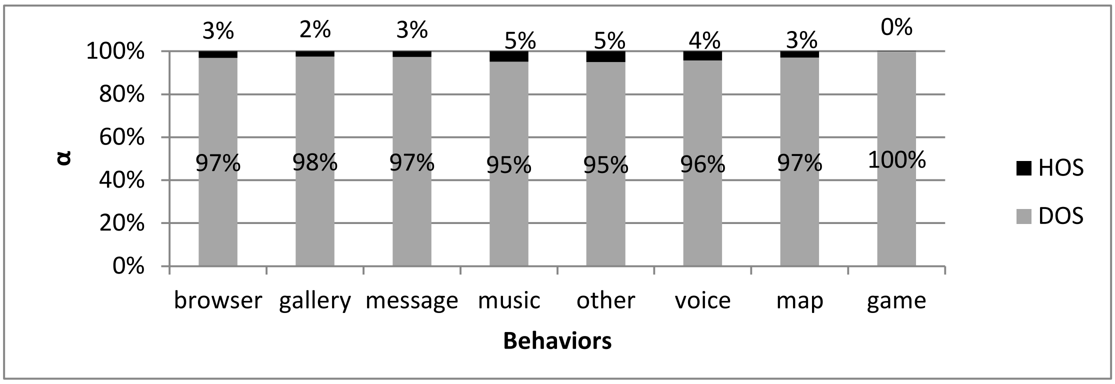


Figure 11 The value of α in different behaviors

5.3 Correlation between BQoSs and BQoE

Given all $BQoS_{i,j}^{k}$ and $BQoE_{i}^{k}$, we aim to find the relationship R_i between $BQoS_{i,j}^{k}$ $(j = 1,2,\ldots,6)$ and $BQoE_{i}^{k}$; that is

$$BQoE_i^k = R(BQoS_{i,1}^k, BQoS_{i,2}^k, BQoS_{i,3}^k, BQoS_{i,4}^k, BQoS_{i,5}^k, BQoS_{i,6}^k).$$

In order to reduce the complexity of R_i, we estimate the relationship between each $BQoS_{i,j}^{k}$. If one $BQoS$ index can dominate another $BQoS$ index, the dominated $BQoS$ index will be removed. In other words, we aim to use as less $BQoS$ indexes as possible to construct the function R_i. In order to estimate the relationship between each $BQoS_{i,j}^{k}$, we adopted coefficient of correlation r, which is determined by

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

where x_u presents a $BQoS_{i,\alpha}^{k}$ sample, y_u is a $BQoS_{i,\beta}^{k}$ 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 functions which are linear function, logarithmic function, exponential function and power function. As Table 5 shows, logarithmic function best fitted our data. We next investigated the logarithmic relationship among $BQoS$s, and see if it is possible can reduce the number of $BQoS$ indexes. A correlation greater than 0.7 may produce the presence of collinearity[21] which may lead to the large standard error. According to our results, 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% shown as Table 6. To avoid the collinearity problem, the final $BQoS$ indexes used to construct the relationship R_i are MaxFI ($BQoS^k_{i,3}$) and NFI ($BQoS^k_{i,6}$). According our result, we can derive that the situations users feel "non-smooth" are divided into the long waiting time and the fragmentary frames. For example, Nexus S smartphone need more time to process the task than Huawei U8860 smartphone in browser behavior under the general CPU utilization. However, the NFI are almost the same, MaxFI index is good for NFI in this situation. On the other hand, users feel "non-smooth" because of the fragmentary frames in other behavior when available CPU utilization is reducing. In that situation, the frame interval is similar so the NFI index is good for MaxFI.

Table 5 The correlation of correlation r between BQoSs and BQoE

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 6 The correlation of correlation r between BQoSs

	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**

5.4 Analysis of the HSER model

For the legal range of $BQoE_i$, we translate $BQoE_i$ the odds ratio of $BQoE_i$with the logarithm function. Given all $BQoS_{i,3}^k$, $BQoS_{i,6}^k$ and $BQoE_i$, we aim to find the relationship R_i among them; that is,

$$log(BQoE_i^k/(1 - BQoE_i^k)) = R_i(BQoS_{i,3}^k, BQoS_{i,6}^k).$$

We adopted the multiple linear regressions to find logarithmic relationship. In order to evaluate the accuracy of the regressions result, we use the coefficient of R^2, which is obtained by

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

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

Table 7 The R Square of the models

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

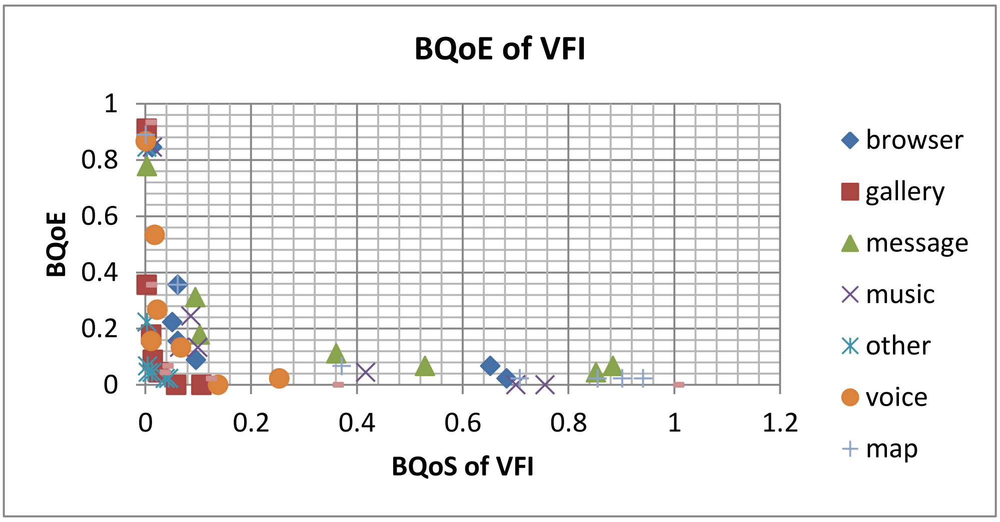


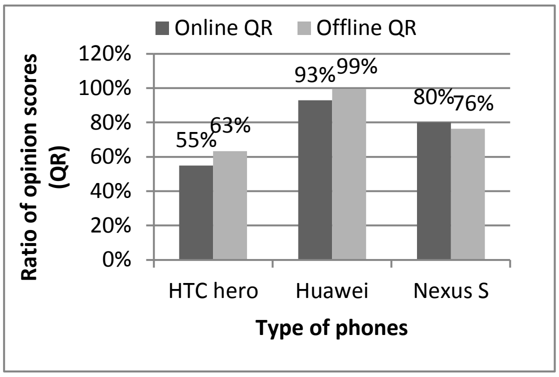
Figure 12 The relationship between the VFI and BQoE

5.5 Evaluation of HSER model

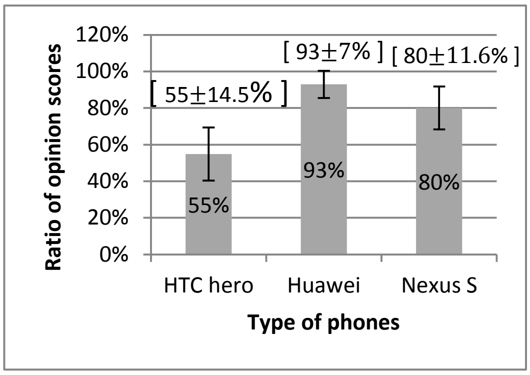
In order to validate our HSER model of different smartphone, we conducted another round of survey. We prepared several video clips about eight operations, shown as Table 4, under normal CPU utilization on three different smartphones. They are HTC hero with 2.2.1 Android platform, Huawei U8860 with 2.3 Android platform, and Nexus S with 4.1.2 Android platform. In this survey, we had 45 volunteers to grade each video with “smooth” or “non-smooth” and used the formula (1) and (2) to compute the questionnaire result, 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). Consider that the online questionnaire have the influence of network delay, we collected 10 volunteers to grade each video with the offline questionnaire. As a result, the influence of the offline questionnaire results, whose ranges are in 95% confidence interval, is lower than 10% shown in Figure 13(a). Therefore, users have good judgment even in the circumstances with network delay. We then adopted 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

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

Our HSER model can have 10% error rate below for each individual behavior.



(a) The satisfaction of smartphones



(b) The confidence interval of smartphones

Figure 13 The satisfaction and confidence interval of smartphones

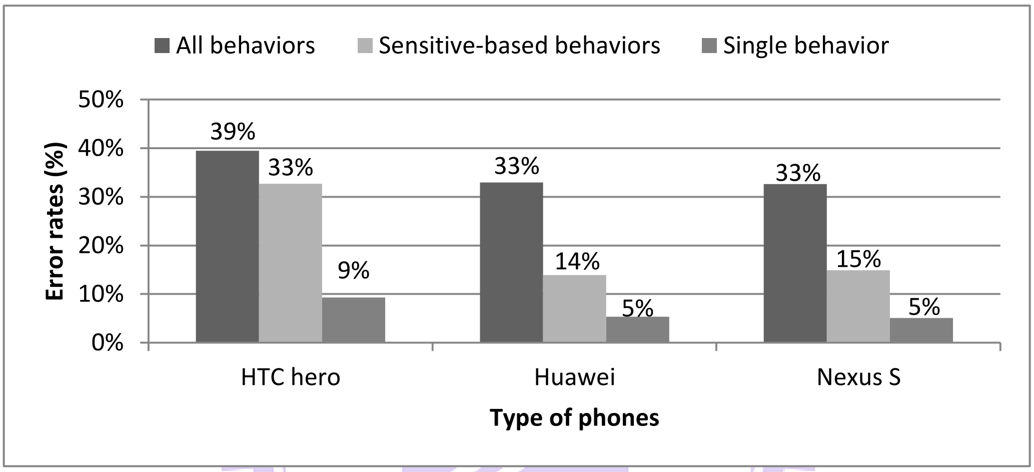


Figure 14 The error rates between the models

Chapter 6 Conclusion and Future Work

In this work, we developed the handheld smoothness evaluation over regression (HSER) model to fairly benchmark the smoothness of smartphones. We first measured BQoS by extracting key 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 frame intervals (NFI). Since the indexes may not always be measurable, especially when the changes between frames are fast, we further developed a tool, named extract device operation sequence (Ex-DOS), to obtain necessary information. Based on obtained behavior-based smoothness quality of services (BQoS), we then designed a questionnaire to determine the relationship between BQoS and behavior-based smoothness quality of experience (BQoE). Finally, we converted the BQoE 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, we conducted several experiments on three different smartphones, HTC hero, Huawei U8860 and Nexus S. We investigated the applicability of the HSER model in different user scenarios. Some user scenarios are timing sensitive while others are not. We validated the correctness of the HSER model by comparing it to our questionnaire results. According to our experiment results, the correlation of MFI, VFI, FNR and TMaxFI is higher than 71.5% in logarithmic relationship. To avoid the collinearity problem, MaxFI and NFI are used to be the indexes for our HSER model. MaxFI and NFI also are good indexes for the "non-smooth" situations of the long waiting time and the fragmentary frames. For individual behavior, the average R^2 is close to 1. In particular, for the behavior of viewing gallery and playing game, the R^2 is up to 0.986 and 0.973. Also, the error rate of HSER is less than 9%. It implies that our regression model can be used to fairly evaluate the smoothness of a smartphone. In addition, the error rate of HTC hero (9%) is higher than other two smartphones (5%). The reason may be the variation that users grade the videos with "smooth" or "non-smooth". The same video for different users will get the different perception.

In the future, we plan to investigate other indexes and collect more users' experience in order to further enhance the accuracy of our model. Possible indexes include the speed of fling and scroll operations. We also plan to improve the accuracy of Ex-DOS tool by detecting non-static objects in a video.

References

[1] Dong Tian, Lansun Shen, and Zhiheng Yao, "Motion Activity Based Wireless Video Quality Perceptual Metric," *Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing*, 2001.

[2] Ruifen Xu, Mingwei Zhang, Lin Jia, and Yongli Li, "A Novel Algorithm for Video Smoothness," *Proceedings of Evaluation Advanced Computer Theory and Engineering*, August 2008.

[3] Rohani Bakar, Muhammad Ibrahim, and D.M. Ali, "Performance Measurement of VoIP over WiMAX 4G Network," *Proceedings of IEEE 8th International Colloquium on Signal Processing and its Applications (CSPA)*, March 2012.

[4] Yu-Chun Chang, Kuan-Ta Chen, Chen-Chi Wu, Chien-Ju Ho, and Chin-Laung Lei, "Online Game QoE Evaluation using Paired Comparisons," *Proceedings of IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, June 2010.

[5] Kuan-Ta Chen, Polly Huang, and Chin-Laung Lei, "Effect of Network Quality on Player Departure Behavior in Online Games," *Proceedings of IEEE Transactions on Parallel and Distributed Systems*, May 2009.

[6] Kuan-Ta Chen, Polly Huang, Chin-Laung Lei, and Chun-Ying Huang, "Quantifying Skype User Satisfaction," *Proceedings of ACM SIGCOMM 2006*, September 2006.

[7] Hyeon-Ju Yoon, "A Study on the Performance of Android Platform," *Proceedings of International Journal on Computer Science and Engineering*, pp. 532–537, 2012.

[8] AnTuTu-Benchmark. [online], available from World Wide Web; https://play.google.com/store/apps/details?id=com.antutu.ABenchMark

[9] SmartBench. [online], available from World Wide Web; https://play.google.com/store/apps/details?id=com.smartbench.twelve

[10] Quantify and Optimize the User Interactions with Android* Devices. [online], available from World Wide Web; http://software.intel.com/en-us/articles/quantify-and-optimize-the-user-interactions-with-android-devices

[11] Miller, R. B., "Response Time in Man-computer Conversational Transactions," *Proceedings of Fall Joint Computer Conference*, 1968.

[12] Jakob Nielsen, "Usability Engineering," published by Morgan Kaufmann, San Francisco, 1993.

[13] Android development. [online], available from World Wide Web;

http://developer.android.com/training/articles/perf-anr.html
[14] Screencast Video Recorder Demo. [online], available from World Wide Web; https://play.google.com/store/apps/details?id=com.ms.screencastfree
[15] Chung-You Wei, Jen-Wei Hsieh, Tei-Wei Kuo, I-Hsiang Lee, Yian-Nien Wu, and Mei-Chin Tsai, "Resource Reservation and Enforcement for Framebuffer-Based Devices," *Proceedings of Ninth International Conference on Real-Time and Embedded Computing Systems and Applications*, 2003.
[16] Yu-Chun Chang, Kuan-Ta Chen, Chen-Chi Wu and Chin-Laung Lei, "A Crowdsourceable QoE Evaluation Framework for Multimedia Content," Proceedings of ACM Multimedia, October 2009.
[17] Android keylogger (AKL). [online], available from World Wide Web; http://www.ebl.org.tw/.
[18] Free Video to JPG Converter Tool. [online], available from World Wide Web; http://www.dvdvideosoft.com/products/dvd/Free-Video-to-JPG-Converter.htm
[19] Nikhil R Pal and Sankar K Pal, "A review on image segmentation techniques," Pattern Recognition 26 (1993) 1277–1294.
[20] H. Verkasalo, "Analysis of Smartphone User Behavior," *Proceedings of IEEE 2010 Ninth International Conference on Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR),* 2010.
[21] Odewale J.O., Agho Collins, Ataga C.D., Odiowaya G., Hamza A., Uwadiae E.O. and Ahanon M.J., "Study of Some Fruit and Seed Traits Relationship and Assessment of Multicollinearity in Date Palm (Phoenix Dactylifera L) Accessions of Nigeria by Correlation and Principal Component Analysis, " *Proceedings of Greener Journal of Agricultural Sciences*, ISSN: 2276-7770 Vol. 3 (2), pp. 164-175, February 2013.