100,000,000 Taps: Analysis and Improvement of Touch Performance in the Large

Niels Henze
University of Oldenburg
Oldenburg, Germany
niels.henze@uni-oldenburg.de

Enrico Rukzio
University of Duisburg-Essen
Essen, Germany
enrico.rukzio@uni-due.de

Susanne Boll
University of Oldenburg
Oldenburg, Germany
susanne.boll@uni-oldenburg.de

ABSTRACT
Touchscreens became the dominant input device for smartphones. Users' touch behaviour has been widely studied in lab studies with a relative low number of participants. In contrast, we published a game in the Android Market that records the touch behaviour when executing a controlled task to collect large amounts of touch events. Players' task is to simply touch circles appearing on the screen. Data from 91,731 installations has been collected and players produced 120,626,225 touch events. We determined the error rates for different target sizes and screen locations. The amount of data enabled us to show that touch positions are systematically skewed. A compensation function that shifts the users' touches to reduce the amount of errors is derived from the data and evaluated by publishing an update of the game. The independent-measures experiment with data from 12,201 installations and 15,326,444 touch events shows that the function reduces the error rate by 7.79%. We argue that such a compensation function could improve the touch performance of virtually every smartphone user.

ACM Classification Keywords
I.3.6 Methodology and Techniques: Interaction techniques; H.5.2 Interfaces and Presentation: User Interfaces - Interaction styles

General Terms
Design, Human Factors, Experimentation

Author Keywords
Touch screen, mobile phone, public study, user study, app store, Android Market, large-scale

INTRODUCTION
Since the introduction of the iPhone, mobile phones with touchscreens began to dominate the smartphone market. Today, all major phone makers have touchscreen devices in their portfolio. Most touchscreen devices have only a few hardware buttons and the touchscreen is used for most input. While touchscreens are well suited for direct manipulation, other use-cases such as text entry using virtual keyboards suffer from the "fat finger problem". Users do not see where they touch and cannot feel the position of virtual keys and buttons. As touchscreens are pervasive and have certain limitations, understanding users' touch performance is crucial to the design of user interfaces for smartphones.

While touchscreens and touch behaviour have been studied for years, understanding even the low-level characteristics of users' touch behaviour remains challenging. If investigating just the error rate for target selection one must not only consider the two dimensions of the screen but also the size of the target, different screen sizes, aspect ratios, tasks, touchscreen technologies and ergonomic aspects. Furthermore, the sequences of touch contacts might affect the touch behaviour and doubles the number of dimensions that must be considered. The error rate is, however, not the only important aspect. While Fitts' law models the time to select a single target, the precision with which targets are hit and the performance for multiple simultaneously presented targets are not sufficiently analysed.

The amount of touch events that can be collected in lab studies to investigate users' touch behaviour might seem high at first glance. Assuming that a single participant produces one touch event per second, 1,800 events can be collected from a single subject in 30 minutes. E.g. Park et al. collected 750 touch events from each participant [9] and Perry et al. collected 1,000 touch event from every subject [10]. But a much higher number of touch events is needed for a detailed analysis, when considering e.g. a touch screen with a resolution of 240x320 pixels that provides 76,800 positions and the fact that usually the effect of the target size has to be tested as well.

In order to collect a truly large amount of touch events, we designed and implemented a mobile game that serves as the apparatus for studying the touch behaviour of smartphone users. In order to attract a large number of participants, the game has been published to the Android Market. Because external factors cannot be ruled out and we have little control over the participants, the study has an inherently low internal validity as there was no possibility to control any contextual factors. The flip side is that the diversity of the environment provides a higher external validity than common lab studies.

MobileHCI 2011, Aug 30-Sept 2, 2011, Stockholm, Sweden.

After discussing related work, we will describe the game that has been developed to collect the data. We provide an overview about the data we collected after publishing the game to the Android Market including the used devices and how much participants played. Following this, an analysis of the touch events is provided that shows the error rate for different target sizes and screen locations. Furthermore, it is shown how touch contacts are skewed relative to the targets position across the screen. The next section reports a compensation function that shifts the touch events systematically to decrease the error rate. We report then how we evaluated this function by publishing an updated version of the game to the Android Market. We show that the function significantly decreases the error rate for the developed game. We close the paper with an outlook to future work and a call for investigating our data in more detail.

RELATED WORK

Research focusing on effective and efficient input and output on mobile devices has always been an important topic as it addresses the inherent conflict between the desire for a small device size and ergonomic aspects defined by the physiology of our eyes, hands and fingers. PDAs being popular in the 90s relied mainly on resistive touchscreens that combined input- and output space in one screen and the user interface was optimized for stylus based input. Most mobile phones used in the beginning a display for output and a physical keypad for input. Nowadays most smartphones rely almost solely on capacitive touchscreens and the users' fingers for input and output. The usage of a stylus came out of fashion as it could easily be lost, it takes time to retrieve them, and their usage implies two-handed interaction. One important aspect to be considered in user interface design is that the output resolution of such a touchscreen is much higher than the input resolution of a human thumb or finger. This leads to the "fat-finger-problem" due to the difficulty to select small targets with a much larger finger and the aspect that the finger occludes the target as well. Current smartphones address this aspect e.g. through a visual confirmation of what has been touched (e.g. the iPhone or Android on-screen keypads show a zoomed-in version of the key currently being touched) or trough callouts that show the region currently touched in order to perform fine granular selections.

One strand of research focuses on interaction techniques which allow the selection of small targets with a finger without changing the size of the target while achieving an acceptable error rate. In Shift [16] this has been achieved through callouts showing a copy of the area occluded by the finger in a non-occluded area and the possibility to move a pointer in the callout via finger movement in order to select the desired target. In Tap-Tap [12] the occluded area is also shown in a callout but here a zoomed in copy of the occluded area is shown and the user has to touch the desired target in the callout with a second touch. In Escape [17] the small targets are visually changed and indicate a direction in which the user has to drag its finger after touching it in order to select it. Close targets indicate different dragging directions through which it is possible to differentiate between several very small and close targets. The disadvantages of those interaction techniques are that additional interactions are required when compared with a simple touch, which requires more time and a higher mental effort.

Further research focused on the optimal size of targets while considering the trade-off between finger size and user interfaces design. For almost perfect accuracy targets would have to have a size of more than 20 x 20 mm [7] which would mean that current touchscreen phones would be able to display only circa 8 targets while showing almost no other information. According to the iOS Human Interface Guidelines [1] the optimal size of a tapable UI element on the iPhone is 6.74 x 6.74 mm which is a compromise between an acceptable error rate and available screen size. There exists a significant body of research that investigated the influence of target size and context on time needed for selecting a target and the error rate [8, 15]. Considered contextual aspects were e.g. the actual task (e.g. inspired by Fitts' law or text input), device- and display-size and -type, thumb size [2], activity (e.g. standing or walking) [13], touch feedback [6] or one-handed or two-handed interaction. The outcome here is often a suggestion regarding an optimal target size and location under consideration of the given context and an assumption regarding acceptable error rate, task load or user satisfaction.

Relatively little research exists that analyses how the actual location of the target on the mobile device or device orientation affects effectiveness and efficiency. Early research focusing on fixed touchscreens mounted on a table showed that users touch slightly below the actual target if the screen is tilted away from the user and that they touch above the target if its tilted towards the user [14]. A further study with a touchscreen tilted towards the user showed that participants not only touch slightly below the target but that there is also an horizontal bias as participant touched on average 0.5 cm left of the actual target [14]. It's been assumed that parallax may explain the vertical bias but no explanation for the horizontal bias has been found.

Previous research showed that the location of targets on the screen has a significant effect on effectiveness, efficiency and user satisfaction. Himberg et al. developed an adaptive on-screen keyboard that observes where the user is touching the display in relationship to the displayed key [4]. This information is used to adapt the shape, size and location of virtual keys in order to improve error rate. Their findings show that right handed participants tended to hit too far on the left and vice versa for left-handed participants. Further studies showed that regions which are easily to reach with the thumb when considering one-handed interaction achieve the best task performance and low perceived difficulty [5]. It has been concluded that frequently used buttons should be placed in those regions. A further study showed again that targets within easy reach of the thumb could be reached very quickly but the accuracy was best when the targets were located on the left, right and top edges of the screen [10]. Park et al. analysed the success rate, error rate and convenience of 25 regions

of a touchscreen when using one-handed thumb input [9]. The authors also analysed the offset between indicated target and actual touch events. They were able to observe location specific offset vectors and discussed the idea of adjusting the location of the touch recognition area in order to improve the overall performance as location dependent offsets.

Our paper is the first to report the horizontal and vertical offset between indicated target centre and actual hit location that is based on a very large data set collected in a realistic context. This allows us in contrast to previous research, that was performed in laboratory settings or which is based on a rather small number of touch events, to calculate those offset vectors very precisely. Furthermore are we the first to show that the consideration of this horizontal and vertical bias could lead to a significant improvement in terms of error rate and precision of target selection.

DESIGN OF THE GAME

In order to collect a large number of touch events and attract a large number of participants we decided to develop a game. During the design we had to find a good balance between providing players with a game that is worth playing and a test application that collects meaningful data. Based on our experience with using Apps as a test platform that is published to a mobile application store (see [3] for an overview) we consider stringent tasks and users' motivation as crucial aspects for the success of this approach in terms of the validity of the data and the widespread usage of the application.

Game play

The game play is inspired by the very controlled task of the study described by Park et al. [9]. Targets are presented to the player and the task is to touch these targets. We decided to use circles as targets as these have the same diameter in all directions and thus allow easier comparison of different target sizes.

The game is structured in three stages called stars, water, and fire. Each stage contains four levels and each level consists of multiple micro levels. In most micro levels, one circle is presented to the player (see Figure 1). The player advances directly to the next micro level without interruption as soon as the target is hit. If a target has not successfully been hit in a certain time frame it is counted as a miss. We used this type of task to enable an analysis that does not need to consider the existence of multiple targets. Every fifth micro levels consists of multiple simultaneously presented targets (see also Figure 1). As soon as a target is successfully hit it disappears. The player must hit all targets to advance to the next micro level.

To make a game out of the two basic tasks they must be challenging for the player. Thus, the player must complete a micro level in a certain timeframe. The timeframe is reduced from micro level to micro level while the player proceeds through a level. The player receives one penalty point if a target has not been hit in this timeframe. The game is lost when the player collected three penalty points in one level. Players receive scores when they successfully hit a target. The faster a target is hit the higher the score.

Different target sizes are used in each level. The average target size is reduced from level to level within each stage. Levels of stage one ("stars") consist of 20 micro levels, in the second stage ("water") the levels have 40 micro levels and the first three levels of the last stage ("fire") consist of 60 micro levels. I order to make it impossible to complete the game, the very last level consist of 80 micro levels and we strongly assume that it is impossible to complete this level. We did this to encourage even very good players to keep playing.

Each stage has a different animated background shown in Figure 1. The background has the purpose to make the game visually appealing but should also increase the external validity because colourful or animated background can be found in many situations (e.g. the Android home screen or mobile games).

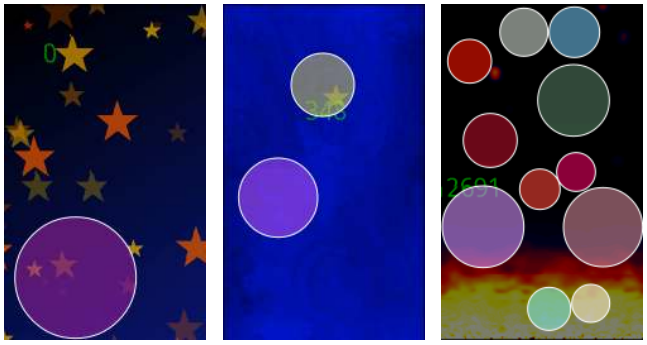

Figure 1. Screenshots of the game's three stages: stars, water, and fire. The left screenshot shows a microlevel with a single circle while the other screenshots show microlevels with multiple targets.

Measures and consent

We collected various data about the characteristics of the used devices and the performance of the players. An unique identifier for each installation is derived from a device's "Android ID" using a hash function. Furthermore, we collect the user's locale (e.g. "en_GB" or "es_ES"), the device's type (e.g. "GT-I9000" for the Samsung Galaxy S), the time zone, and the resolution. We log the position and size of the targets for each micro level while the user is playing. We also log the position of each touch event and the time elapsed since the start of the micro level. The device's orientation (provided by the phone's accelerometer) is recorded for each touch event but was logged only for 88.63% of the data because we had not integrated this measure in the first version of the game.

The properties of the used device are transmitted to our server when the game is started. The data collected while playing is transmitted after a level is completed no matter if it was successfully completed or not. The data is stored internally on the phone and retransmitted after the next level is completed if the transmission fails.

We do not collect any data that allows identifying individual players or installations. We decided to clearly inform players about the fact that data is collected in

order to act ethically (see [3, 11] for a discussion) and to conform to corresponding legislation in many countries. The modal dialog shown in Figure 2 tells players that they are about to participate in a study when the game is started for the first time.

Motivation

We tried to make the game visually appealing in order to motivate intensive usage. We integrated different animated backgrounds and a star highlights the score a player received when a target is successfully hit. The total sore is also shown in the background and continuously moves across the screen to minimize its effect on a player's average performance. Furthermore a player receives a "badge" when successfully completing a level. To increase the long term motivation we implemented a global and a local high score lists shown in Figure 2. If a twitter client is detected on the player's device a "tweet this" button allows players to share their score each time they achieve a high score.

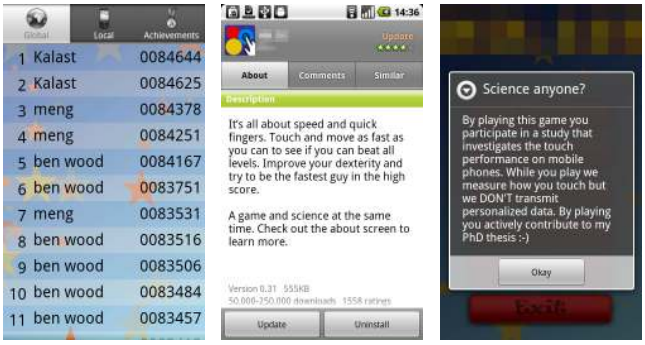

Figure 2. The game's high score list (left), description in the Android Market (middle), and the modal dialogue that informs player about the conducted study (right).

PUBLISHING IN THE MARKET

We published Hit It![1] in the Android Market on October 31, 2010. The description of the game is shown in Figure 2. Till January 22, 2011 the game was installed 108,987 times according to Google's Developer Console. The game received 1,666 ratings with an average 4.19 on the five point scale (the higher the better) provided by the Android Market and there are 640 comments about the game in the Android Market. In total we collected data from 99,749 installations but only 91,731 installations provided meaningful data (see below). We cannot control who contributes to the results as players installed the game at their own will. We provide an overview about the data in the following.

Devices, resolutions, and locales

In total we collected data from devices with 321 different model names. Most of these model names appeared, however, only a few times and have are rather exotic name such as the "I-Mobile i858 Jackbox Edition". For 93 model names we collected data from more than a hundred installations. As the mobile network operators give different model names to the same device type

[1]Hit It! in the Android Market: `https://market.android.com/details?id=net.nhenze.game.button2`

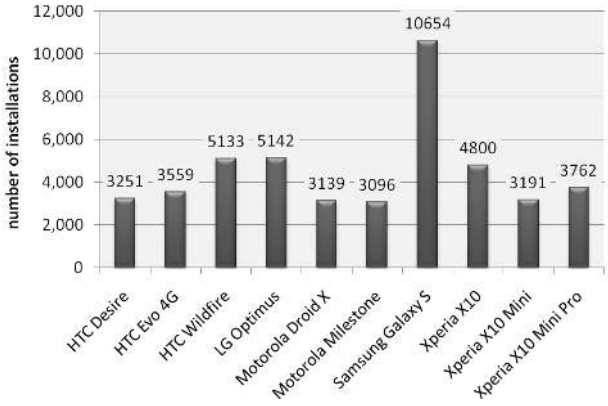

Figure 3. The ten most common devices.

there are in fact less different devices than these 93 names suggest. The Samsung Galaxy S, for example, appears with at least seven different model names. We harmonized the model names for common devices. The ten most common devices which represent 49.85% of all installations are shown in Figure 3.

Looking at the device's resolution is important as it can be used to identify the aspect ratio which we considered in our analysis. In total we found 20 different resolutions but 13 resolutions are used by less than 500 installations. The resolutions that are used by more than 500 installations are shown in Figure 4. The four most common resolutions are 320x240, 480x320, 800x480, and 854x480. For each of these four resolutions we collected data from more than 15,000 installations.

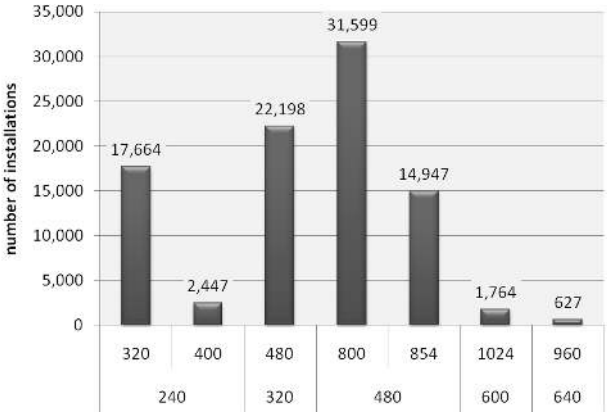

Figure 4. Resolutions with more than 1000 installations.

The collected locales and time zones show that there is a strong bias towards western countries among the players. In particular, the two most common locales are English speaking US ("en_US", 48.31%) and English speaking Great Britain ("en_GB", 9.68%). This is followed by France ("fr_FR", 4.81%), Japan ("ja_JP", 3.71%), Taiwan ("zh_TW", 3.18%), and Germany ("de_DE", 3.00%). The other 190 locales together result in 27.32% including 56 further English locales representing 3.91% of all installations. The selected time zones show a similar picture. The only non US American or European time zone among the ten most common time zones is Asia/Tokyo.

Amount of collected data

While we received data from 99,749 installations not all of them provided meaningful data. We only use data from 91,731 installations because we either did not record a single played level or because the data is inconsistent due to multiple installations that share a single identifier (a known bug of the Android platform).

In total 1,950,888 levels have been played and on average 21.27 levels (SD=66.99) have been played on each installation. On 45.16% of the installations less than 10 levels have been played. There are, however, a few very intensive players and ten installations, for example, contributed more than 2,000 levels each. Figure 5 provides an overview about the number of played levels per installation. The number of collected touch events per device is analogue. In total 120,626,225 touch events have been recorded and on average 1,315 (SD=5,575) events have been produced on each device.

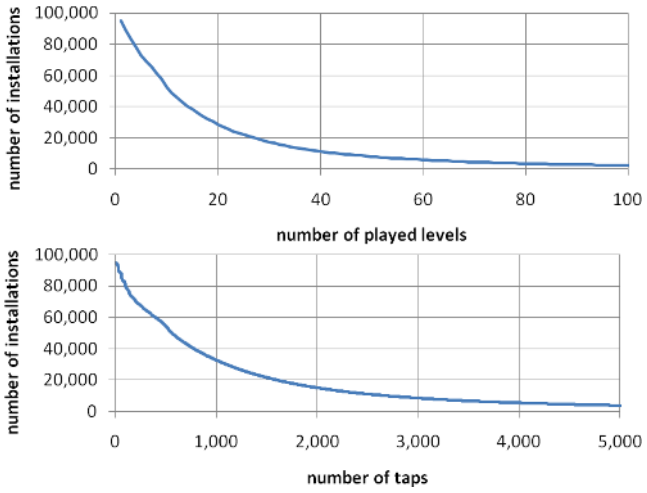

Figure 5. Number of played levels and produced touch events. Both graphs must be interpreted as on y installations more than x levels have been played. E.g. on 30.000 installations more than 20 levels have been played.

ERROR RATES

We analysed the collected data to investigate the relation between the targets' size and the error rate as well as the targets' position and the error rate. An error occurred when the user touched the screen outside of a displayed target. Both aspects are crucial when designing the interface for touchscreen devices because interface designers have to find a balance between the amount of information that should be displayed and the size of touchable interface elements. As the screen size as well the device's form factor influences the error rate we treated the devices separately. In the following we describe the results for the most common device for each of the four most common resolutions: the HTC Wildfire, the LG Optimus One, the Samsung Galaxy S, and Sony Erricson's Xperia X10.

Errors per target size

Only micro levels with a single target are considered too analyse the relation between target size and error rate. We removed the very first played level from each result set to reduce noise and only the first eight levels are considered because the last stage ("fire") contains an unbeatable level. Furthermore, only players that contributed at least a hundred micro levels are contained in the results to reduce the amount of noise in the data. Through this we considered 1,766,207 touch events for the HTC Wildfire, 1,827,444 for the LG Optimus One, 3,810,875 for the Samsung Galaxy S, and 1,366,672 for Sony Erricson's Xperia X10.

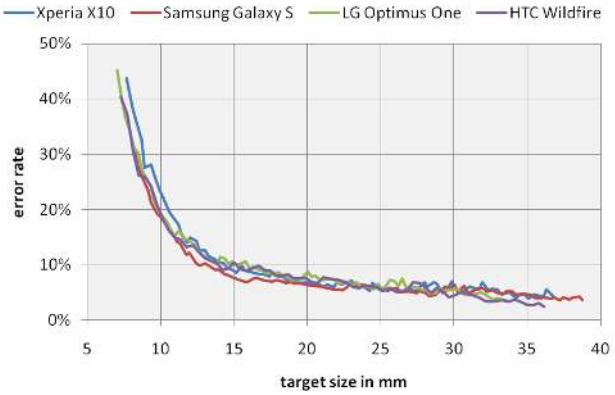

Figure 6. Ratio of errors for different target sizes. The diagrams show the percentage of errors for the Xperia X10, the Samsung Galaxy S, the LG Optimus One, and the HTC Wildfire.

As the four devices have different screen size and resolutions the target sizes have been normalized to millimetre. Figure 6 shows the average error rate for the target sizes. As one would expect the error rate decreases with an increasing target size.

Errors per position

To analyse the relation between targets' positions and the error rate we divided the screen in 10x10 areas. For each of these areas we determined the average error rate for all targets whose centre falls in the area. As large targets do not fit in the areas at the border we only consider targets smaller than 12mm in order to not disadvantage areas at the border. Furthermore, we focused again on the most common device for each of the four most common resolutions. As only targets smaller than 12mm are used we only considered 311,492 touch events for the HTC Wildfire, 404,943 for the LG Optimus One, 552,889 for the Samsung Galaxy S, and 221,879 for Sony Erricson's Xperia X10. The results are shown in Figure 7. Apparently the error rate at the border is much higher than in the centre. Looking, for example at the Wildfire the average error rate at the border is 31.68% but 17.59% in the centre.

Discussion

The error rates show the same trend across the four devices. Below 15mm the error rate dramatically increases and jumps to over 40% for targets smaller than 8mm. These results cannot be directly compared with previous research as we use circles instead of squares as discussed previously. Our observed error rates are higher than error rates reported by [8, 10] when assuming that the circles' diameter is equivalent to a quadrate's diameter. Here one has to consider that the area of a circle is

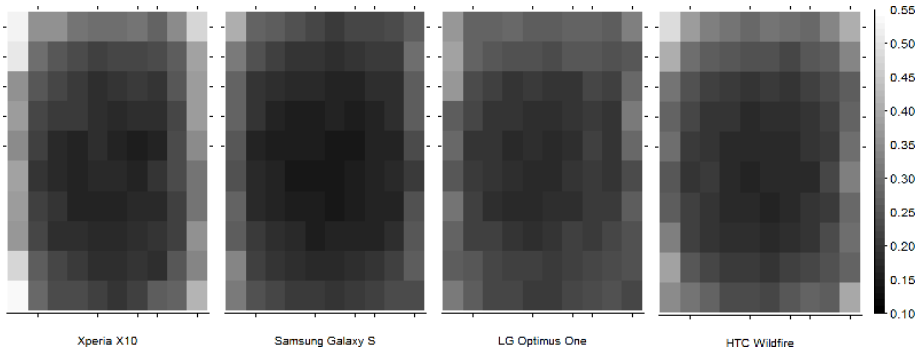

Figure 7. Ratio of errors across the display for targets smaller than 12mm (dark areas indicate regions with a low error rate and bright areas indicate regions with a high error rate).

larger than the area of the quadrate when assuming that both have the same diameter. For a diameter of 10.89mm (a quadrate with an edge length of 7.7mm) Parhi et al. found an error rate of 5% [8], Perry et al. found an error rate of 12.3% [10], while we observed an error rate of 14.19%-18.86%. We assume that the higher error rate can be explained by our participants' higher time pressure and because they were not able to see the next target in advance. Furthermore, the animated background might impact the players' performance.

In contrast to [10] we found that especially the border of the screen seems to be more difficult to hit than the centre. The results obviously do not directly generalise to other use cases as players had to touch targets under high temporal pressure. We assume that the relative differences in the error rates should, however, be transferable.

From those findings we conclude that the error rate for a target could be reduced be enlarging it and by moving it towards the centre of the screen. Furthermore, one could consider the relationship between the error rate for different target sizes and different screen positions. The difference between the border of the HTC Wildfire (31.68% error rate for targets smaller than 12mm) and the centre (17.59% error rate for targets smaller than 12mm), for example, corresponds to an increase of the target's size from 8mm to 11mm.

TOUCH POSITIONS

We further analysed the collected touch events to investigate if a systematic offset between the targets' position and the touched positions exists. We derive a compensation function that tries to reduce the number of errors and we evaluate this function by simulating the effect on collected data that wasn't used for deriving the function.

Touch offsets

To demonstrate the systematic skew we determined the offset between each touch and the according target for all micro levels with a single target. Figure 8 and Figure 9 exemplary show the distribution of the deviations along the x-axis and the y-axis for the HTC Wildfire and four selected areas considering on average 129,138 touch events for each region from 5,133 installations in total. When discussing offsets then we assume that the origin of the used X/Y coordinate system is in the top-left corner.

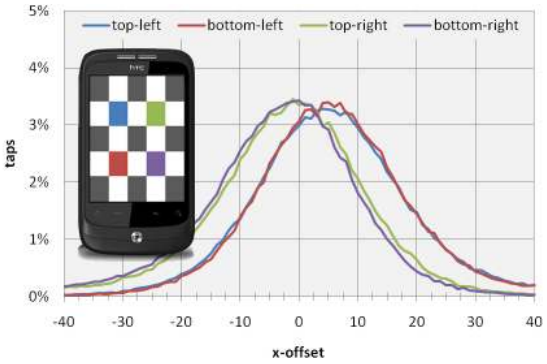

Figure 8. Histograms of touch offsets along the x-axis for the HTC Wildfire. Each line visualizes the touch distribution for targets in the coloured rectangles shown in the checkerboard pattern.

A positive x-offset indicates that the touched location is too far on the right and a positive y-offset indicates that the touched location is too far to the bottom. The offsets discussed in the following show the distance between the centre of the circles and the actual touched location. The average offset of the top-left area (marked in blue) is x=4.90px and y=6.64px (SD(x)=11.91, SD(y)=12.01). That means that the touches are shifted towards the lower-right corner of the screen. In contrast, the average offset of the bottom-right area (marked in purple) is x=-2.55px and y=-2.24px (SD(x)=11.63, SD(y)=12.57), which means that the touches are slightly shifted towards the top-left corner of the screen.

To estimate the offset across the screen we divided the screen in 10x10 areas. For each area we determined the average offset of the touch events for all targets whose centre is inside the area along the x-axis and the y-axis. As large targets do not fit in the areas at the border we only consider targets smaller than 12mm. Furthermore, we only consider touch events that hit the target to avoid shifting the average offset towards the centre of the screen. The x-offset and the y-offset are than used to determine an offset vector. The orientations of these vectors and the vectors' length for the Samsung Galaxy S are shown in Figure 10 for which we could consider 440,004 touch events from 8,201 installations.

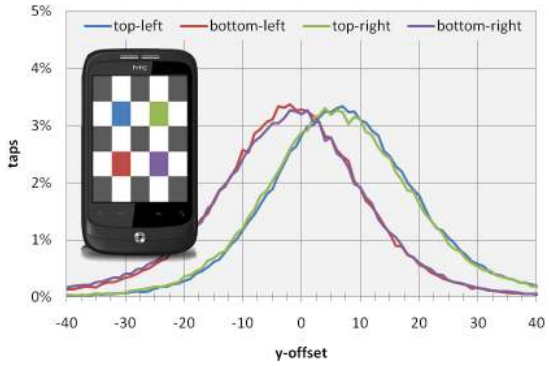

Figure 9. Histograms of touch offsets along the y-axis for the HTC Wildfire. Each line visualizes the touch distribution for targets in the coloured rectangles shown in the checkerboard pattern.

The touch events for the Samsung Galaxy S are systematically skewed towards a position in the lower-right of the screen and the offset vectors point towards this position. We approximated the exact location by testing the vector field for every screen position. The position with the lowest average deviation from the empirically determined vectors is at the position 358px/605px. This position is 1.33cm (≈25% of the screen's width) left and 2.12cm (≈25% of the screen's height) above the lower-right corner of the screen. Looking at the other devices shows a similar skew. For the HTC Wildfire the position is approximately 1.22cm left and 2.44cm above the lower-right corner of the screen. The touch events for the LG Optimus One are skewed towards 0.90cm left and 1.35cm above the lower-right corner of the screen and for Sony Ericsson's Xperia X10 it is 1.49cm left and 1.77cm above the lower-right corner of the screen.

For all four phones the touch events are systematically skewed. Furthermore, the observed effect is consistent with the preliminary data presented in [9]. As only hit targets that are smaller than 12mm are considered the offset is likely even larger for bigger targets. The direction of the overall skew suggests that the touch events are shifted towards a position where the user's thumb naturally touches the screen if the phone is held in the right hand. Unfortunately we do not know how participants held their phone while playing. In particular, it would be interesting to compare right-handed and left-handed users because it could be assumed that the offset is mirrored along the y-axis for left-handed users.

Shifting touch positions

As shown in the previous section, a systematic offset between the targets' centre and the touched position exists. As the offset is systematic it might be possible to apply a function to the touch events that compensates this offset. Based on the 75% of the considered data we derived a function that tries to minimize the number of errors. Therefore we divided the screen in 30x30 areas. For each area we determined how touch events that fall in this area should be moved to reduce the total number of errors for this area. Only micro levels with one circle are considered. In contrast to the previous section, hits and misses as well as targets of all sizes are used.

As we aim at evaluating the function on a large basis we do not differentiate between individual devices. As the four most common resolutions have different aspect ratios we, however, treat each resolution individually. For each resolution and each of its areas we tested all possible shifts. To determine the shift vectors we considered 6,373,962 touch events and 17,467 installation for 240x320, 8,597,786 touch events and 21,199 installation for 320x480, 12,431,333 touch events and 31,473 installation for 480x800, and 5,078,545 touch events and 14,917 installations for 480x854. Thereby, we determined one vector that shifts the touch event for each area. The shifts along the x-axis and y-axis for the Samsung Galaxy S are shown in Figure 11.

Applying the determined shift vectors on the remaining 25% of the data that has not been used for training leads to a reduction of the total number of errors of 3.13% for installations with 240x320 pixels, of 3.18% for installations with 320x480 pixels, of 3.49% for installations with 480x800 pixels, and of 4.08% for installations

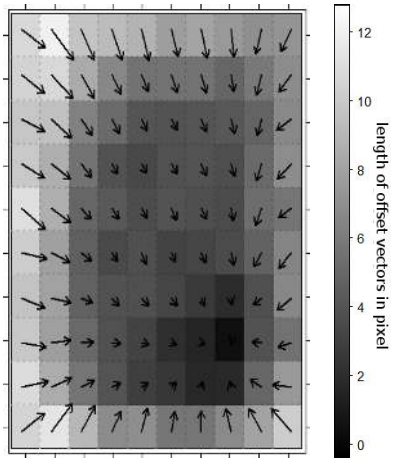

Figure 10. Orientation and relative length of the offset vectors for the Samsung Galaxy S.

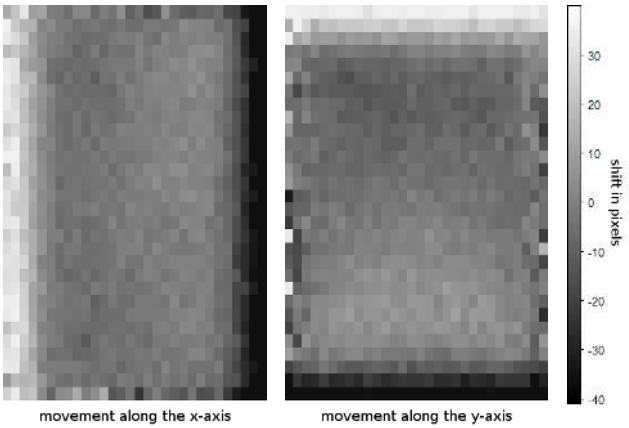

Figure 11. Emperically determined values to move touch events along the x-axis (left) and the y-axis (right) for a resolution of 480x800 pixels.

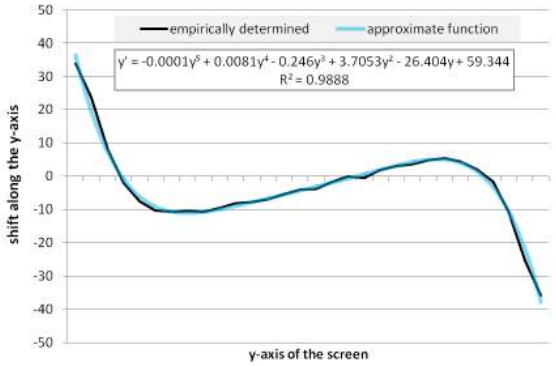

Figure 12. Average shift along the y-axis and the approximate polynomial function for a resolution of 480x800 pixels.

with 480x854 pixels. On average, the number of errors is reduced by around 3.5%.

Analysing the shift vectors we suppose that the determined y-shift only depends on the y-value and the x-shift only depends on the x-value. Therefore, we computed the average y-shift values over the complete width of the screen. Furthermore, we derived a polynomial function that approximate the curve to determine a continuous compensation function. The curve and its approximate function is shown in Figure 12 for the 480x800 resolution.

We accordingly processed the data for the x-shift and the other resolutions. Again we used the 25% of the data that has not been used for training to test the determined compensation functions. They reduce the number of errors by 3.51% for installations with 240x320 pixels, by 3.42% for installations with 320x480 pixels, by 3.65% for installations with 480x800 pixels, and by 4.59% for installations with 480x854 pixels. The number of errors is further reduced for all resolutions compared to using the raw values and results in an average error reduction of around 3.8%. We assume that two factors contribute to this improvement. Instead of discrete vectors that treat all touch events in one area equal, a continuous function is used. Thereby, each screen position is treated differently. Furthermore, the individual shift vectors are affected by noise. The influence of this noise is reduced by using the average of all x-shifts and y-shifts along the respective axis.

EVALUATION

We integrated the previously discussed compensation function in our game in order to investigate its effect on the users' performance. By publishing an update of the game we conducted an independent measures quasi-experiment with one independent variable.

Design

In the previous section we analysed how the compensation functions perform when applied to the remaining 25% of the touch events not used to determine the function. The simulation can, however, not take into account if players adapt to the function. Therefore, an independent-measures experiment was conducted. The compensation function described in the previous section was integrated in the game for the four resolutions 240x320, 320x480, 480x800, and 480x854. Each installation is randomly assigned to either the control condition or to the experimental condition when the game is started for the first time. In the control condition, no function is applied to the touch events while in the experimental condition the compensation function is used to shift the touch events. Players are neither informed that their touch events are shifted nor about the condition they are assigned to.

The error rate as well as the distance between touch events and targets is used as dependent variables. Two types of errors contribute to the error rate: Whenever a player's touch misses a circle and whenever a circle is not hit within the given time frame. The number of errors is divided by the number of presented circles resulting in the error rate. A player's average distance between touch events and the targets' centre is measured only for touch events that hit a target in order to rule out the effect of the error rate on this measure. Our hypothesis was that the error rate will be lower and the distance will be smaller for all resolutions and all played levels. We assumed that the improvement for the error rate will be at least as high as the results of the simulation.

We deployed an update of the game to the Android Market on January 22, 2011. All players that updated or newly installed the game are considered in the following. As players are free to play or stop at any time the study is not a true experiment but a quasi-experiment.

Results

For the analysis we considered only the four most common resolutions (240x320, 320x480, 480x800, 480x854) representing 94.16% of all installations of the updated game. The remaining data consist of data from 7,847 installations for the control condition and 7,910 installations for the experimental condition. We removed the first ever played level from the data provided by each device to further reduce the noise in the data. Furthermore, we only consider data sets that contribute at least a hundred circles to the respective analysis. Through this we were able to consider 7,435,733 touch events from 6062 installations for the control condition and 7,890,711 touch events from 6139 installations for the experimental condition in total. A one tailed dependent t-test for unequal variance is used to test our hypothesis.

Comparing the installations in the control condition (n=6062) with the installations in the experimental condition (n=6139) after shaping the data as described shows that the manipulation had a significant effect ($p<.0001$, $r=0.04$) on the error rate. The error rate for the control condition ($M=13.39$, $SD=14.81$) is significantly higher than for the experimental condition ($M=12.34$, $SD=13.83$). The error rate per circle for the experimental condition is 7.79% lower than for the control condition (see also Figure 13).

To make the distance between touch events and targets' centres comparable across different resolutions the distance is converted from pixels to percentage of the screen's width. The applied function also had a significant effect on the distance between touch events and the targets' centres ($p<10^{-27}$, $r=0.10$). In the control condi-

tion (n=5921) the distance was 2.93% larger (equivalent to just one pixel on a Samsung Galaxy S) than in the experimental condition (n=5993).

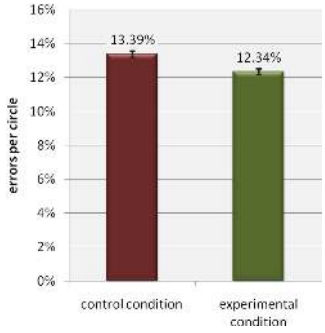

Figure 13. Overall error rates for all levels from installations with one of the addressed resolutions (error bars show standard error).

We further analysed the data treating the four resolutions independently to see if the function reduces the error rate for all resolutions (see Figure 14). The error rate is reduced by 10.69% for 240x320 ($n_{cont.}$=1005, $n_{exp.}$=1050), by 2.38% for 320x480 ($n_{cont.}$=1600, $n_{exp.}$=1638), by 7.53% for 480x800 ($n_{cont.}$=2396, $n_{exp.}$=2403), and by 6.17% for 480x854 ($n_{cont.}$=1061, $n_{exp.}$=1048). The difference is significant for 240x320 (p<.01, r=0.06) and 480x800 (p<.01, r=0.03) but not for 480x854 (p=.10, r=0.06) and also not for 320x480 if we consider a reduced significance level (p=.02, r=0.04).

The control condition's average distance between touch events and the targets' centre is significantly higher than the experimental condition's distance for all resolution with a small effect (p<.00001, r=0.1) but for 480x854 where the manipulation had a very small effect (p<.01, r=0.06).

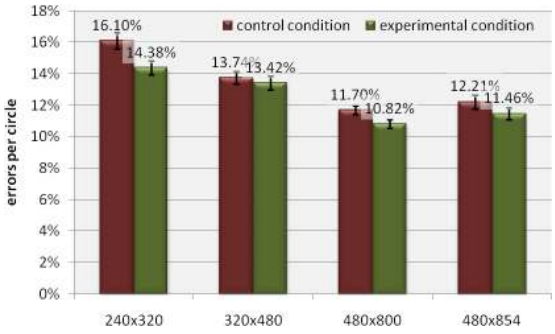

Figure 14. Error rates for all levels from installations with the respective resolution (error bars show standard error).

Looking at the four first level the error rate is reduced for all resolutions and all but the first level of the 320x480 resolution ($n_{cont.}$=273, $n_{exp.}$=278) where the error rate is increased by 4.98% percent (p=.31, r=-0.02). Figure 15 exemplarily shows the error rate for 240x320. For this resolution the error rate is reduced by 19.39%-24.20%. The differences are significant only for the third (p=.01, r=0.14) and fourth level (p=.01, r=0.14) with a small effect size. The distance between touch events and the targets' centre is lower for all resolutions and all of the first four levels. The average distance is reduced by 0.44%-5.52%. The difference is significant at a .01 level for 9 out of 16 levels. The manipulation had a small effect for 9 and a very small for 7 of the 16 levels.

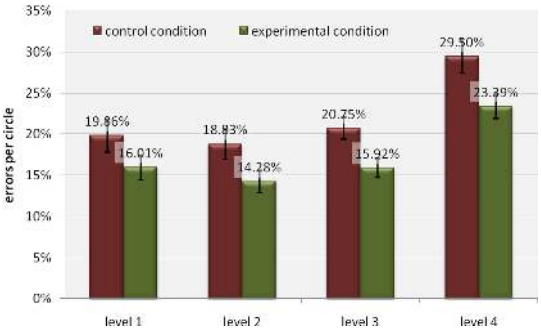

Figure 15. Error rates for installations with a resolution of 240x320 for the first four levels (error bars show standard error).

Discussion

The evaluation shows that the applied function significantly reduces the error rate if considering all resolutions. The error rate is also reduced for the four resolutions but the difference is not always significant. Even if looking at the individual levels for each resolution shows that the error rate is lower for all but one of the 16 analysed levels. We assumed that the reversed result for this level is by chance. The effect size, however, is very small for the overall result and the individual resolutions. It is still small if treating the levels separately. Similarly the distance between touch events and the targets' centre is reduced for all analysed levels. Overall, the difference is significant and we observed a small effect and also for most analysed levels we determined a significant small effect.

We assume that the applied function improves the performance of the average players in all circumstances. The function does not only reduce the error rate but also improves the players' precision. The small or very small effect size, however, shows that only a small percentage of the variance can be explained by our manipulation. Looking again at the data returned by individual installations we see some abnormal high error rates. For some installations the error rate is higher than 80% for both conditions. These devices strongly influence the effect size and we have no explanation for them. The differences between the resolutions, devices, and levels also add variance to the data and reduce the effect size.

In general we assume that the developed function improves the players' performance. As the function does not differentiate between different devices with the same resolution the error rate might be further reduced if the functions are constructed for individual devices. One hint that supports this assumption is the small improvement for the resolution 320x480. For this resolution we found not only more different devices but we also assume that the differences between the devices are larger than for the other resolutions. As the developed functions

treat the x-axis and the y-axis separately this leaves further room for improvement.

This evaluation cannot show that the developed function would improve users' touch performance for other tasks. We, actually, assume that this specific function would not improve the users' performance for a wide range of tasks because it is optimized to reduce the error rate for one specific task. A function that just cancels out the systematic skew described in the previous section could already be more general. Incorporating data for other tasks and probably also incorporating more degrees of freedom such as the device's orientation, handedness of the user and the timing could result in a generally applicable function that improves the touch performance. Deriving functions for specific tasks and widgets is another option. This evaluation, however, does show that a function to improve users' touch performance exist at least for one tasks. The function's simplicity makes it plausible to assume that functions for other tasks and probably also a generally applicable function exists.

SUMMARY AND FUTURE WORK

In this paper we analysed the touch behaviour of smartphone users. Using a game published to the Android Market we collected more than 120 million touch events from 91,731 installations. We used the data to determine the error rate for different target sizes and screen positions. The data is also used to show that touch events are systematically skewed toward a position in the lower-right of screen. Based on this finding we trained a function that shifts the touch events to reduce the number of errors. The function is evaluated by publishing an update of the game to the Android Market. We showed that the function significantly reduces the players' error rate and improves their precision across different resolutions and devices.

Our findings have a rather low internal validity as we cannot control how the users play the game but our results have a very high external validity due to the large number of contexts, users and devices which have been considered. A further aspect to be taken into account is that we considered touch events recorded during a game play which stays in contrast to the rather formal tasks performed in [8, 9]. As we focus on the basic characteristics of the touch behaviour we nonetheless strongly assume that the results are meaningful beyond the scope of this specific game. A further limitation is the self-selectiveness of the participants. We attracted participants that like to play this particular kind of game. Compared to lab studies our approach has the advantage that the participants are more diverse and despite a bias towards western countries come from all over the world.

The results presented in this paper are only a fraction of what can be investigated using the collected data. We would like to share the data with other to enable further analysis[2]. We are particularly interested in investigating touch sequences. A touch event might not only depend on the target's size and location but also on previous touch contacts. We plan to apply the findings presented in this paper and also findings about touch sequences to virtual keyboards. As we showed that the error rate can be reduced for the game we hope that we can similarly reduce the error rate for virtual keyboards.

[2]Data is available from `http://nhenze.net/?page_id=673`

Acknowledgments: Parts of this work were conducted within the context of the Emmy Noether research group Mobile Interaction with Pervasive User Interfaces funded by the German Research Foundation (DFG).

REFERENCES

1. Apple Inc. iOS Human Interface Guidelines, 2010.
2. V. Balakrishnan and P. Yeow. A study of the effect of thumb sizes on mobile phone texting satisfaction. *Journal of Usability Studies*, 3(3), 2008.
3. N. Henze, M. Pielot, B. Poppinga, T. Schinke, and S. Boll. My App is an Experiment: Experience from User Studies in Mobile App Stores. *accepted by the IJMHCI*, 2011.
4. J. Himberg, J. Hkkil, P. Kangas, and J. Mntyjrvi. On-line personalization of a touch screen based keyboard. In *Proc. IUI*, 2003.
5. A. Karlson. Interface Design for Single-Handed Use of Small Devices. In *Proc. UIST*, 2008.
6. S. Lee and S. Zhai. The performance of touch screen soft buttons. In *Proc. CHI*, 2009.
7. J. R. Lewis. Literature review of touch screen research from 1980 to 1992. In *IBM Technical Report 54.694*, 1993.
8. P. Parhi, A. Karlson, and B. Bederson. Target size study for one-handed thumb use on small touchscreen devices. In *Proc. MobileHCI*, 2006.
9. Y. Park, S. Han, J. Park, and Y. Cho. Touch key design for target selection on a mobile phone. In *Proc. MobileHCI*, 2008.
10. K. Perry and J. Hourcade. Evaluating one handed thumb tapping on mobile touchscreen devices. In *Proc. GI*, 2008.
11. M. Pielot, N. Henze, and S. Boll. Experiments in App Stores - How to Ask Users for their Consent? In *Proc. workshop on Ethics, logs & videotape*, 2011.
12. A. Roudaut, S. Huot, and E. Lecolinet. TapTap and MagStick: improving one-handed target acquisition on small touch-screens. In *Proc. AVI*, 2008.
13. B. Schildbach and E. Rukzio. Investigating selection and reading performance on a mobile phone while walking. In *Proc. MobileHCI*, 2010.
14. A. Sears. Improving touchscreen keyboards: Design issues and a comparison with other devices. *Interacting with computers*, 3(3), 1991.
15. A. Sears and Y. Zha. Data entry for mobile devices using soft keyboards: Understanding the effects of keyboard size and user tasks. *International Journal of Human-Computer Interaction*, 16(2), 2003.
16. D. Vogel and P. Baudisch. Shift: a technique for operating pen-based interfaces using touch. In *Proc. CHI*, 2007.
17. K. Yatani, K. Partridge, M. Bern, and M. Newman. Escape: a target selection technique using visually-cued gestures. In *Proc. CHI*, 2008.