Analysis and Modeling of Grid Performance on Touchscreen Mobile Devices

Ken Pfeuffer[1,2]*, **Yang Li**[1]
[1]Google Research & Machine Intelligence, Mountain View, CA, Unites States
[2]Lancaster University, Lancaster, United Kingdom
k.pfeuffer@lancaster.ac.uk, yangli@acm.org

ABSTRACT
Touchscreen mobile devices can afford rich interaction behaviors but they are complex to model. Scrollable two-dimensional grids are a common user interface on mobile devices that allow users to access a large number of items on a small screen by direct touch. By analyzing touch input and eye gaze of users during grid interaction, we reveal how multiple performance components come into play in such a task, including navigation, visual search and pointing. These findings inspired us to design a novel predictive model that combines these components for modeling grid tasks. We realized these model components by employing both traditional analytical methods and data-driven machine learning approaches. In addition to showing high accuracy achieved by our model in predicting human performance on a test dataset, we demonstrate how such a model can lead to a significant reduction in interaction time when used in a predictive user interface.

ACM Classification Keywords
H.5.2. Information Interfaces and Presentation (e.g. HCI): Theory and methods.

Author Keywords
Grid UI; touchscreen mobile device; performance modeling; machine learning; predictive interfaces.

INTRODUCTION
Touchscreen smartphones have become the predominant computing devices today, with rich user experience afforded by direct touch and high mobility of these devices. However, interaction behaviors on these small-form factor devices are difficult to analyze and model. Studying and modeling these behaviors allow us to better understand how users behave and to establish a computational basis for optimizing user interfaces. User performance modeling is a core foundation of HCI. For example, researchers have explored fundamental models for pointing [26], steering [2], menus [19], and visual search [5, 24]. These efforts aim to predict the difficulty of a task, such as the time required to accomplish the task, without running expensive user studies, and reveal new possibilities to create and improve user interfaces.

However, the majority of prior modeling work focused on desktop computers, leaving mobile devices surprisingly underexplored despite their large adoption into everyday life. For instance, menu selection is an important interaction task that has been extensively studied for a desktop environment [7, 12, 15, 19, 25]. While prior models may be adapted to a mobile context [9], mobile device provide dramatically different interaction affordances. A desktop menu can show all of its items at once that might only occupy a small part of a large monitor screen. In contrast, a mobile menu can only show a fraction of its items while consuming the entire mobile screen. In addition, mobile devices allows direct touch input, which contrasts cursor movements in desktop menus.

In this work, we focus on a scrollable two-dimensional grid UI, a common interface on touchscreen mobile devices for presenting a large number of items on a small-form factor device, e.g., organizing apps in a launcher or photos in a gallery. The task itself, which is to find and select an item by a tap, is rather simple to the user. However, modeling this behavior is highly complex, which involves performance components such as navigation, visual search, pointing and learning, as well as the interplay among them. While many existing theories and findings are relevant, there lacks a holistic understanding about user behaviors and performance for interaction with a grid interface and a computational model for capturing many nuances when using a touchscreen smartphone.

To better understand factors that might be relevant, we first present an analysis of a 20-user study of mobile grid interfaces. The study reveals how components such as learning, navigation, visual search and pointing affect the user's performance in response to varying grid sizes and trial repetitions. The analysis resulted in novel findings about mobile grids. For example, users switch between two navigation strategies. Users become faster over time by optimizing gesture operation and visual search. We also discovered how human performance depends on row regions of a grid.

Based on the findings from the study, we propose a predictive model for human performance in a grid task. A unique aspect that our model captures is the probabilistic determination of user navigation strategy, as users decide between starting at

*The work was conducted when the author was an intern at Google.

CHI 2018 April 21–26, 2018, Montreal, QC, Canada

ACM ISBN 978-1-4503-5620-6/18/04.
DOI: https://doi.org/10.1145/3173574.3173862

the top or bottom of the grid. Conditioned on the strategy, the cost of navigation is linear with the row position of the target item. Once the viewport is set, users visually search across columns, and point and touch the target. Because after scrolling the target's on-screen position for computing pointing cost is unknown, we combine Fitts' law cost with a probability distribution of position across the screen, by learning from the data. Initial evaluation based on how well our model predicts human performance on a test dataset shows promising results on predictive quality. Our paper offers the following contributions:

- Discovered novel findings about user behaviors on scrollable grid interfaces by conducting a user study on touchscreen mobile devices.
- Proposed and evaluated a predictive model for human performance that brings together new empirical findings, existing theories and machine learning methods, which adds to the knowledge of task modeling for mobile interfaces.
- Demonstrated how a performance model can be used for utility-based UI optimization via a theoretical simulation involving factors such as task distribution and complexity and prediction quality.

RELATED WORK

Models of menu performance can be divided into two categories: simulation and mathematical models [6, 7]. Simulation models consider visual search as a main component to explain the factors that influence human performance. Simulation models include EPIC [24] and ACT-R/PM [12], which are based on a production-rule architecture, and differ in how visual search is conducted. EPIC is based on serial and parallel visual processing of menu items; ACT-R/PM is based on top-to-bottom visual search and considers coordinated perception, action and cognition. These models provide an overlying theory for visual search behaviors, but do not fit well to icon search [20], and may not characterize well user behavior [13].

Mathematical models are based on non-linear regression equations to predict total selection time as a function of task and user characteristics. They incorporate findings from studies that found menu selection depending on factors such as menu length (a longer menu requires more time [30]), organization (e.g., alphabetic order is faster than random order [15, 27, 28]), target position (targets at a top area are easier to reach than the rest in the menu [13]), and practice (faster with practice [19]).

For example, Cockburn et al. introduce the SDP model that predicts selection time with three components: visual search, decision, and pointing, which are modeled by a linear function with the item number, the Hick-Hymen Law [21], and Fitts' Law [26], respectively. SDP also accounts for the transition from novice to expert users by applying an expertise factor that modulates based on the amount of prior experience with the target. Bailly et al. predict menu performance by a probability density distribution of the user's gaze across the UI [7]. Their model includes serial and directed visual search, and with practice a user transitions from the former to the latter behavior that improves selection time. This phenomenon is due to the user being able to remember the spatially stable target positions and directly glance at them [37].

Scrolling navigation is another aspect commonly found in menus, particularly on mobile devices with small screens. Scrolling navigation affects performance due to the need to actuate scrolling manually and the involvement of the perceptual, cognitive and motor resources [39]. Researchers explored models of scrolling, proposing a linear function of distance to the target when the target is unknown [4] and logarithmic otherwise [22] as confirmed in Cockburn and Gutwin's studies [17]. In the latter work, they also extend SDP with a scrolling component by iterative reapplication of search, decision, and pointing tasks. However, some of the assumptions do not hold for mobile interfaces, e.g., users do not scroll via a scrollbar or always navigate top down.

In general, navigation tasks can be divided into the two stages: navigation and pointing [16, 33]. Because after navigation the starting position for pointing is unknown, the spread of end points can be modeled with a Normal distribution [14]. Researchers have proposed scrolling models that include multiple stages (for each scroll action) on indirect input devices [16] and for absolute rate-based scrolling on large touchscreens [40]. Our work adds to the knowledge of accelerated scrolling behaviors [32] by studying direct touch-based grid navigation.

Several previous efforts have studied grid UIs. Cockburn and Gutwin explored the CIM model for constrained interfaces such as grids [18]. However, the grid in their work is operated by buttons to control a cursor, instead of using direct touch. Ahlström et al. investigate Square Menus [3] on desktops. Studies indicated higher expert performance than traditional linear and pie menus, where pointing distance is reduced and the time for visual search was found linear with the number of items. CommandMaps [34] is another grid variant for desktop UIs modeled by SDP, which shows all the command layers at once (instead of just one), taking advantage of the user's spatial memory to improve selection.

Overall, while previous work has studied many performance building blocks, our work differs in that we focus on mobile grids by combining multiple performance components and addressing specificity of touchscreen mobile devices. This is challenging due to the interplay between these components and increased uncertainty in user behaviors. In the following, we provide an in-depth study of an mobile grid interface to gain a fresh view on the subject, and discuss new findings.

USER STUDY

We conducted a study to investigate how users find and select targets from a grid UI that involves a large number of items. From the study, we intend to better understand factors that influence the user's behavior, which eventually inform our design of a predictive model for human performance.

Grid Selection Tasks

In this study, a grid interface shows a collection of mobile app icons in a two dimensional, row by column organization. A selection task involves 1-DOF vertical scrolling and tapping to select that users commonly perform on touchscreen mobile

devices. If the target is within the initial (top) viewport, a user can directly tap on it without scrolling.

We implemented the study software based on Android in Java using the default gesture behavior that includes flick-down to the end. Graphical design such as the item size was adapted from the Android App Launcher. Irrelevant UI elements such as system notification bars or navigation buttons were hidden to avoid distractions, which provides the full screen for the grid UI. Depending on the viewport position, there are 6-7 rows of the grid visible to the user at a time (Figure 1b). After selecting an app icon, its background flashes in either green or red to indicate a correct or wrong selection to the user.

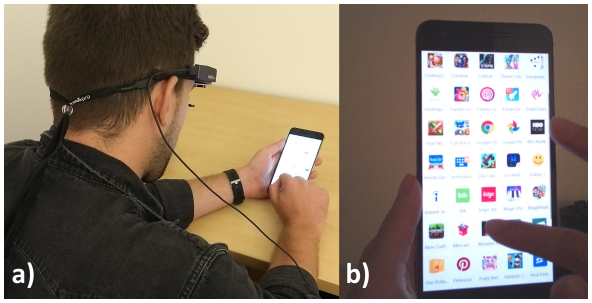

Figure 1. The experimental setup (a) and a close-up view of the grid interface (b) in the study.

Study Design

We employed a within-subject repeated measures study design. The main condition is *Gridlength* as the number of rows in the grid, varied among 12, 18, 24, and 30 rows with a counterbalanced order. With the fixed number of columns 5, there are 60, 90, 120, and 150 items in these grids, which resemble realistic scenarios of grid UIs, e.g., there are often 80-100 apps on average in an app launcher [11, 38]. We select the items for each grid randomly from the top 420 apps of the Android Play Store in July 2017, and ensure each app is used at most once for each user to eliminate potential carry over effects.

We used *8 blocks* per gridlength to investigate learning effects. Each Length×Block condition consists of *15 trials* (order randomized). We ensured at least one trial for every second row to cover the whole menu length in each block. For example, for gridlength=30 rows, users select one target every second row. For gridlength=12, six targets were fixed and remaining targets randomly added to reach 15. The column position was randomly selected from a uniform distribution. The same set of targets are used across blocks for a gridlength.

In sum:

20 Users ×
4 Grid Length (12-, 18-, 24-, 30-row) ×
8 Blocks ×
15 Trials
= 9600 trials.

Participants and Setup

20 paid participants took part in the study. They were 31 years old on average (SD=5), 8 female, all right handed and active smartphone users. Their background is IT student or employee in a large IT company. Users rated their experience with the scrolling grid UI for applications from 1 (no experience) to 5 (frequent daily use) with M=3.1 (SD=1.5). We used a Nexus 6P mobile phone that comes with a 5.7" 2560×1440px display, 159.3×77.8×7.3mm size, Android OS, 3GB RAM, and a Qualcomm Snapdragon 810 CPU. Users were instructed to hold the device in their left hand and touch with their right. Figure 1 shows the setup and a user during the study.

We used a Tobii Glasses 2 eye tracker to collect gaze data, running at max. 100Hz. The tracker provides a live video feed of the user's view and gaze during the study for observation and post-hoc video analysis. The tracker gets calibrated with a 1-point calibration, conducted at the beginning of the study. It was repeated when a custom-build 9-point accuracy test visibly showed a poor calibration on the live gaze video feed. On average, the accuracy was 1.8° visual angle (SD=3.2°), or 259px (SD=446px) considering an average eyes-to-display distance of 40 cm. 4 users had to be excluded from gaze analysis as the tracker software reported invalid trials, otherwise a valid gaze sample rate of 94.7% was reported. We consider the data as not qualified for research on fixation level but sufficient for sampled analysis across display areas.

Users were first briefed and filled out a consent and a demographic form. Then the eye tracker was worn and calibrated. Users were introduced to the study application, UI aspects such as scrolling direction and alphabetical sorting, and the task goal (to be fast and error free). No training was conducted as users were intuitively familiar with the basic task and UI. Next, users started the experiment that is a sequence of 32 blocks. A break followed after each block with at least 10 seconds enforced by the study tool. Users were free to take longer a break but typically started immediately after the 10 seconds. Before each trial, the application icon and name were shown for the user to memorize.

Data Preprocessing and Statistical Method

Click error rate (wrong target selected) was low at 2%. For the subsequent analysis, we exclude all click errors and outliers (time > 3 SDs from the mean). With this criteria, 3.7% (358/9600) of trials were excluded. For statistical analysis, we employ independent ANOVA tests with Greenhouse Geisser and Bonferroni correction using SPSS's General Linear Model/Repeated Measures. Error bars in the figures are 95% CI and correspond to the statistical analysis.

RESULTS AND ANALYSIS

We first examine the overall aspects of grid tasks: learning effect and performance with each grid length. We then analyze the three performance components of a grid task: navigation, visual search, and pointing.

Learning and Menu Performance

User expertise increases with training (Figure 2a). A main effect was found with task completion time × Block ($F^{2.6}_{49.9}=192.8$, $p<.0001$). Time decreases with an increasing amount of blocks, showing that users learn and become faster over time. All pairwise comparisons were significant ($p<.006$) except for comparisons between block 5-6, 6-7, 6-8, and 7-8

($p>.05$). In Figure 2a, results indicate that learning converges after 5-6 repetitions. The learning curve depicts a negative exponential decrease, aligning with prior menu studies [7, 19], to model with the power law of practice [29].

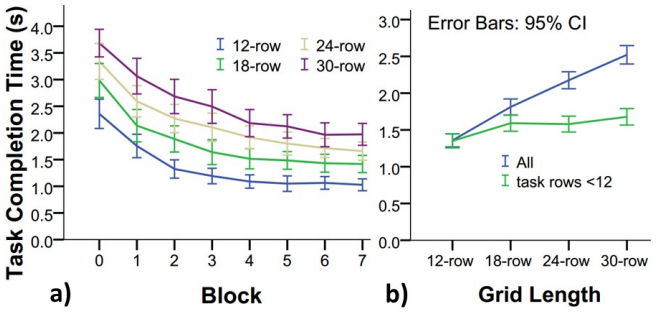

Figure 2. Task completion time for Grid Length×Block, and Grid Length considering two cases.

Longer menus require more time to complete (Figure 2b, Blue). A main effect was found for task completion time × Gridlength ($F^{3}_{57}=73.7$, $p<.0001$). All pairwise comparisons were significant ($p<.01$) indicating a linear increase of time with increasing length. An interaction effect was found for Block×Length ($F^{6.8}_{129.4}=2.2$, $p=.04$) but no pairwise comparisons were found as significant.

Same target positions in a longer grid need more time (Figure 2b, Green). We investigated Nilsen's menu length effect [30] replicated in Bailly et al.'s study [7], where same target positions require more time when the menu is longer. As data, we selected the first 12 rows that are shared by all menu lengths. A main effect on task completion time × Length was significant ($F^{3}_{57}=5.2$, $p<.004$). Pairs 12-row *vs.* 18-row ($p=.03$) and 12-row *vs.* 30-row ($p=.001$) were significantly different. Figure 2b shows the times, indicating longer grids need more time than the 12-row one, but the effect diminishes when tasks involve scrolling navigation over larger grids.

Navigation

There can be no navigation at all when the target lies within the initial viewport position. In all other cases, however, navigation takes up the major portion of a grid task, involving visual and cognitive processes in coordination with the user's precise manual scrolling activities. We first provide information about learning scrolling, and then inspect grid position effects.

Learning Gestural Navigation (Figure 3)

The statistical analysis of Block showed that users become faster over time. We investigated how users navigate faster with an analysis of the amount of gestures involved, start time, speed, and duration. For statistical analysis, we computed the average for each condition (Gridlength×Block). Note that the analysis of the gesture amount and start time is based on all the trials, while gesture speed and duration analysis only uses trials in which users scrolled. If a trial had multiple scroll gestures, their average was used. This led to the following findings that characterize how users optimize their manual touch interactions, and partially explain the user's learning effect. Over time, users:

- performed less gestures ($F^{2.8}_{52.2}=24.5$, $p<.0001$),
- initiated gestures earlier ($F^{1.9}_{35.6}=87.3$, $p<.0001$),
- dragged with a higher speed ($F^{3.5}_{67.6}=7.4$, $p<.0001$), and
- dragged with a shorter duration ($F^{3.6}_{69.3}=27$, $p<.0001$).

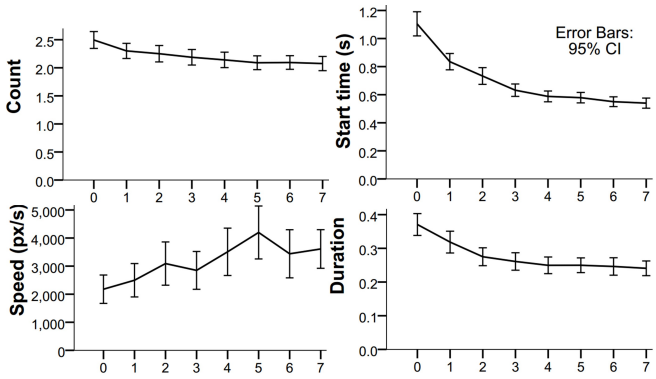

Figure 3. Gesture characteristics for each condition across blocks.

Grid Position Effect

To inspect the effect of position on navigation, we first look at the task completion time across row and column dimensions of the grid. We select the data based on the respective dimensions. For rows, we limit the analysis to every second row to cover an equal range across rows.

Task completion time×column is shown in Figure 4a, and did not reveal statistically significant differences ($F^{4}_{76}=.8$, $p=.5$). This is expected as the column size is fixed, and users navigate across rows, although there seems to be slight tendency towards the center rows.

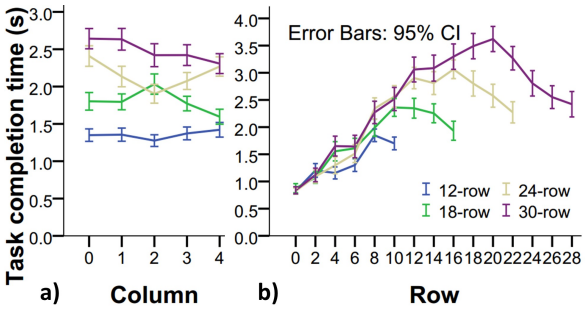

Figure 4. Task completion time across row and columns of the grid.

Across rows, time increases and later decreases (Figure 4b). The task completion time×row effect showed statistical significance ($F^{4.4}_{84}=52.4$, $p<.0001$). The curves show navigation is neither linear nor logarithmic as reported in previous scrolling studies [4, 22]. In particular, task completion time initially increases with row, but also steadily decreases towards the end. This is similar to Bailly et al.'s *last item effect* finding [7]: the last items show decreasing times. However, their menu was a linear, non-scrollable desktop menu, while our grid task includes navigation as a major task component.

Two strategies (Figure 5). We hypothesize that navigation strategy is the main reason for the decrease of the time required, in particular the following two strategies (% show frequencies):

1. **Top-down (80.2%)**: The user navigates from the top of the grid continuously downwards, until the target is found.
2. **Bottom-up (19.8%)**: The user performs a flick gesture to scroll to the bottom of the grid. Then, the user navigates up and selects the target.

We split the data by the two strategies to detail this issue. We analyzed scrolling gestures of the users based on a simplified condition. If the user's first scroll gesture led to reaching the bottom of the grid, we label the trial as bottom-up strategy, otherwise as top-down. This way, overall the bottom-up strategy was used 19.8% in the study. Notably, 12.6% of the bottom-up trials included upward scrolling after the bottom reach (across all trials: 2.5%). One can see in Figure 5 that with both strategies, time increases approximately linear with row (decreasing for bottom-up), and the 30-row condition shows signs of logarithmic increase. In essence, the bottom-up strategy resembles the inverse of the top-down strategy plus initial scroll-down gesture. The average of both strategies explains the increasing/decreasing slopes displayed in Figure 4b.

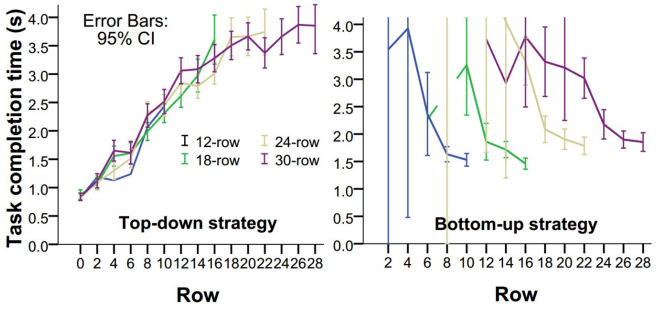

Figure 5. Task completion time for across Row×Grid, for each strategy.

The first letter of the target name affects strategy usage (Figure 6a). The first letter of the target item name is an important clue for the user to guess where the item is roughly located in an alphabetically sorted grid and thus decide which strategy to use. Figure 6a shows that as the initial letter is positioned towards the end of the alphabet, the user tends to use the bottom-up strategy more often. For most letters, users went for the top-down strategy. The relationship between initial letters and strategy usages approximates a sigmoidal curve.

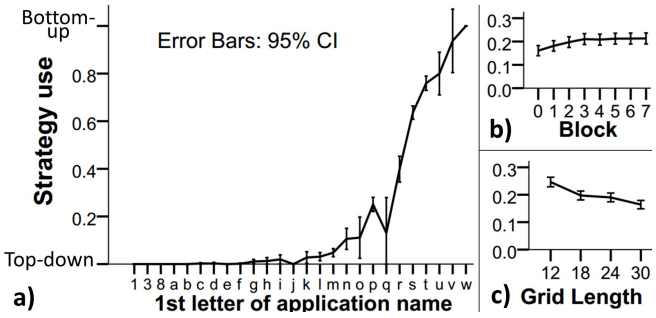

Figure 6. Navigation strategies. The Y-axis shows the chance that the bottom-up versus the top-down strategy is used for different initial letters of a target item.

Experience increases the bottom-up strategy use (Figure 6c). Users employ the bottom-up strategy slightly more often with increasing Block (F^7_{133}=9.4, p<.0001). Pairwise comparisons show the first block is significantly different to blocks 3-7 (p<.008). This is because in the first block the user is not aware of the grid structure, thus cannot efficiently utilize the strategy—however this changes with experience.

Shorter grid length encourages the bottom-up strategy use (Figure 6d). We found a significant effect of grid lengths on strategy usage (F^3_{57}=4.2, p=.01). Only pair 12-row *vs.* 30-row was significantly different (p=.04), indicating that the short 12-row grid leads to a more frequent bottom-up strategy use. This is probably due to the 12-row grid allowing to view the second half of all items when scrolling to the bottom.

Visual Search

Visual search can play a major part in menu performance [13], which users can optimize with increasing knowledge about the spatially consistent menu parts [23, 35]. The grid UI includes spatial consistency (e.g., columns) and inconsistency (the absolute row position of a target changes after scrolling). We analyze user gaze frequency that reflect visual search effort for both dimensions.

Columns: users tend to look at the middle (Figure 7). We analyzed gaze frequencies per column across blocks are shown for all trials (a) and, as an example, for trials where the target was in column 0 (b). The distribution of each block is represented in a different color. (a) shows that gaze points mostly lie within the center column, and linearly decreases towards the sides, aligning with the tendency of task completion time decreases as the column position is towards the center. With increasing blocks, users spent less time looking at the middle and more time at the sides. Thus, users may memorize the column position of a target item and direct their gaze more equally to the target. (b) shows how the gaze frequencies per column change over time. Initially, about 20% gaze data were on the target column, but with training it increases to about 60%, and frequency of other columns gradually decreases.

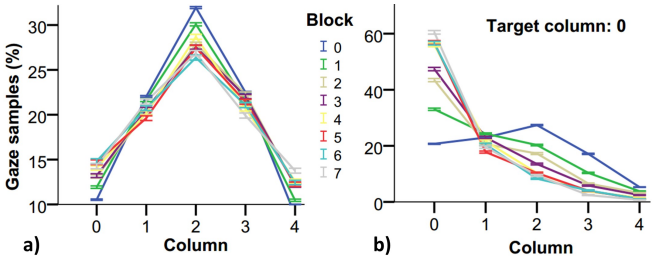

Figure 7. Gaze frequencies per column, normalised.

Rows: gaze is more frequent at initial and final (target's) rows (Figure 8). As the rows are spatially inconsistent, we plot gaze distribution task-specific similar to Figure 7b. It shows that there are typically two larger areas users gaze at (and visually perceive), the initial viewport area (rows 0-6) also known as the pre-gaze search [7], and the target row area. For example, when the target is at row 18, the second peak of gaze samples is found in an area around this position. This reaffirms Bailly et al.'s ideas, as after users navigate to an area including the

target, the user then spends time for visual search and the eventual pointing task centered around the target row position.

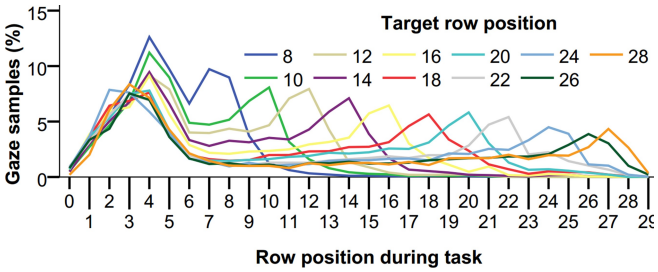

Figure 8. Gaze frequencies per row, for a set of tasks defined by the row position of the target.

In the following, we regard *gaze-to-target distance*, a metric used to understand visual behavior for goal-oriented tasks [10, 31, 36]. We treat the horizontal and vertical gaze-to-target distance separately to investigate column and row, respectively. The *horizontal distance* is the distance of gaze X and target X coordinate. For *vertical distance*, target Y is not given as it changes at scroll, so we use the Y coordinate of the target after the user's last scroll gesture. For each block×grid condition, we average all trials' averaged distances.

With a shorter grid row length, users look more to the target row (Figure 9a). Grid length significantly affected the vertical gaze-to-target distance ($F^{1.4}_{20.8}$=16.3, p=.0003). This is likely due to shorter grids having a higher ratio of spatially consistent-to-inconsistent rows. No effect was found for gridlength on horizontal distance (F^{3}_{45}=2.1, p=.12).

With experience, users look closer to the target (Figure 9b). Block significantly affected horizontal ($F^{3.7}_{55.8}$=83.4, p<.0001) and vertical gaze-to-target distance (F^{7}_{105}=12.6, p<.0001). A logarithmic decrease is revealed, as observed in the prior analysis of learning (Fig. 2a) and scrolling gestures (Fig. 3). The horizontal is visibly closer to the target than the vertical, aligning with the prior result on gaze frequency.

Users look closer when rows are at top (Figure 9c). We split gaze-to-target distance by row areas to further explore the disparity of the Y- to the X-distance. We see for the first fixed 6 rows, Y-distance approximates X-distance, but increases with the rows that require scrolling. Users optimize their visual search for the columns, but not for rows except those at the top of the grid, as they are spatially consistent (in the absolute screen coordinate) with the initial viewport position.

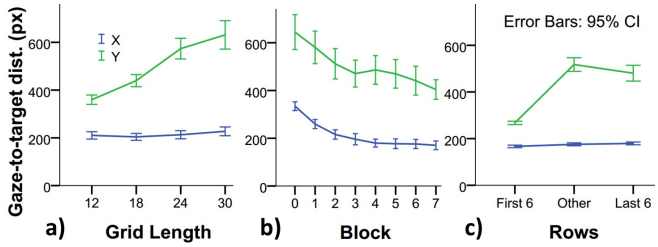

Figure 9. Gaze-to-target distances across grid lengths, blocks and row positions.

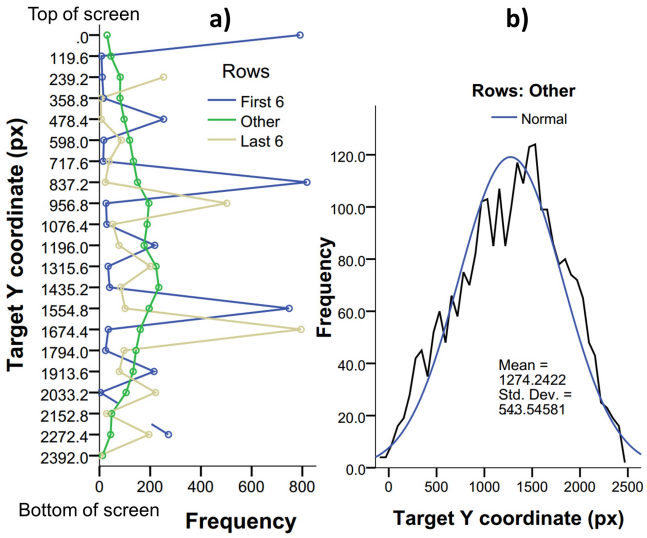

Figure 10. The distributions of the absolute Y positions when target items are in the top and bottom viewpoint (a), and in other parts of the grid (b).

Pointing

The pointing sub task can be considered to be well modeled by Fitts' Law as shown in previous work [9]. The main input parameters are the width of the target and the distance from the starting position of the finger to the target position. The former is constant in a grid, as item size. The latter poses two challenges. The starting position of the finger [40] and and the absolute Y position of the target on the screen are unknown.

The absolute Y position of a target item after scroll is normally distributed around the screen center (Figure 10b). When a target item is within in the top or the bottom viewpoint, i.e., among the first or the last six rows, the absolute Y position tends to be at a specific location (Figure 10a). For other cases where item positions are more affected by scroll, the target position is indeed skewed towards the center of the screen, which we speculate is easier to reach with the finger (Figure 10b). A similar distribution was observed by Cao et al.'s for dynamic peephole pointing tasks [14].

Users seemed faster at the lower part of the screen (Figure 11b). When a trial involves scrolling, we found it is generally faster for a user to acquire the target when it ends up at the lower portion of the screen after scroll (Figure 11b). We speculate that this is because there are more downwards navigation cases than upwards ones and the lower portion of the screen is easier to reach during downwards navigation. There is no clear pattern when the target item is in the first and last 6 rows of the grid (Figure 11a and c). When there is no scrolling occurred (Green plots), users directly touched the target, which were generally faster than those with scrolling (Blue plots).

Users look in the direction they scroll (Figure 11d). To further investigate the above observation, we analyzed the eye tracking data. We hypothesize that users are faster at lower areas because they also look at the lower area when scrolling down. As a result, targets in this area are visually acquired faster,

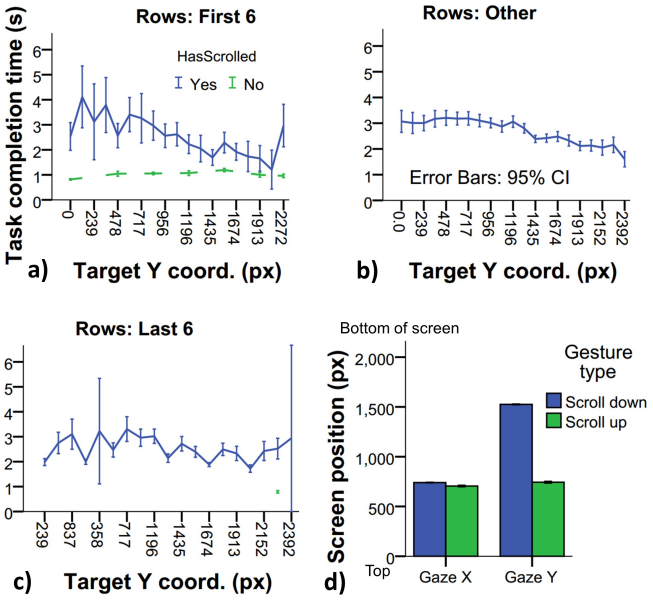

Figure 11. Task completion time plotted against the absolute Y position of the target on the screen (a-c), and the user's average gaze positions during scrolling gestures (d).

hence faster selection. Figure 11d shows the user's average absolute gaze position on the screen while performing scrolling down and up gestures. The data confirms our hypothesis, as the Y coordinate of gaze is clearly at the lower area of the display when scrolling down, and vice versa.

MODELING

Based by the findings from our study, we propose a predictive model of grid performance that integrates the effort for navigation, visual search, and pointing. Our model is a nonlinear function including analytical, probabilistic, and machine-learned components. It has the following input variables:

- len_{row}: the number of rows in the grid.
- len_{col}: the number of columns in the grid.
- $view_{row}$: the length of the viewport in number of rows.
- pos_{row}: the row position of the target.
- pos_{col}: the column position of the target.
- l: the first letter of the target's name.
- w: the width of an item.
- t: the number of previous encounters with the target.

The overall selection time T_i for each item is the sum of navigation T_{nav}, visual search T_{vs}, and pointing time T_{point}:

$$T_i = T_{nav} + T_{vs} + T_{point} \quad (1)$$

Navigation

The navigation time results from a probabilistic combination of the top-down (N_{tdn}) and the bottom-up (N_{bup}) navigation costs:

$$T_{nav} = (1 - S_{btm})N_{tdn} + S_{btm}N_{bup} \quad (2)$$

where S_{btm} is a probability that the user navigates from bottom.

Navigation Costs

The time for both navigation costs can be modeled with a linear function of the target's row position. The difference is that they use a different intercept term, because the bottom-up navigation involves the effort to reach the end of the grid, i.e., the initial swipe down. The bottom-up navigation also counts the row position from the bottom upwards instead of from the top downwards:

$$N_{tdn} = pos_{row}T_{row} + b_{tdn} \quad (3)$$

$$N_{bup} = (len_{row} - pos_{row})T_{row} + b_{bup} \quad (4)$$

where T_{row} denotes the time required for a each row, and b_{tdn} and b_{bup} are bias terms to be learned from the data. To take into account the effect that users become better at navigating the rows with practice, T_{row} incorporates the learning rate that decreases logarithmically with experience, modeled by the power law of practice, inspired by Bailly et al's model [7]:

$$T_{row} = a_r exp(-b_r t) + c_r \quad (5)$$

where a_r, b_r, and c_r are parameters to be learned.

Regulating Strategies

The strategy switch model has three cases determined by the grid length and viewport size, i.e., the top rows, in-between, and the last rows in the grid:

$$S_{btm} = \begin{cases} 0 & \text{if } pos_{row} < view_{row} \\ 1 & \text{if } pos_{row} > len_{row} - view_{row} \\ S_{prob} & \text{otherwise} \end{cases} \quad (6)$$

where S_{prob} is a sigmoidal function that outputs a probability between 0 and 1, based on a linear combination of three values: the gridlength, the user experience, and the first letter of the target name:

$$S_{prob} = sigmoid(s_0 + s_1 len_{row} + s_2 S_{exp} + s_3 l) \quad (7)$$

We normalize len_{row} and l to ± 0.5 by considering the range of these values in the dataset: $len_{row} = len_{row}/30 - 0.5$ (our study dataset has ≤ 30 rows) and $l = l/36 - 0.5$ (10 digits + 26 letters for names). We used the sigmoid function because it approximates the data distribution well (Figure 6) and offers a suitable numerical range of 0 to 1 for regulating navigation cost. S_{exp}, the expertise of using a strategy, is computed as follows:

$$S_{exp} = sigmoid(e_0 + e_1 t) \quad (8)$$

which is a sigmoid function based on a linear transformation of the previous encounters t. The sigmoid function maps the count that is an unbounded value to a range between 0 and 1. Sigmoid is appropriate in that expertise will eventually saturate with practice which will approach 1 infinitely. s_i and e_i are the parameters to be learned from the data. The entire strategy model is equivalent to a standard feed-forward neural network using sigmoid as the activation function, and these parameters can be learned using general stochastic gradient descent (more details in the next section).

Visual Search

Users tend to visually attend to the center area more frequently than border regions, and from there search to the side. Based

on this observation, we model visual search as a linear scan from the center of the columns:

$$T_{vs} = |len_{col}/2 - pos_{col}|T_{col} + v \quad (9)$$

where v is the bias term, and T_{col} is the time for the user to visually scan each column. As with experience, users remember the column positions and direct their gaze more efficiently. Thus, it integrates the power law of practice:

$$T_{col} = a_{vs}exp(-b_{vs}t) + c_{vs} \quad (10)$$

where a_{vs}, b_{vs}, and c_{vs} are parameters to be learned.

Pointing

The absolute position of a target on the screen that is needed for Fitts' law calculation is undetermined due to the scrolling behavior. To address this issue, we use a probabilistic combination of Fitts' Law efforts across all vertical positions. We estimate the target's Y position on the screen using a Gaussian distribution as inspired by the study finding (see Figure 10), while X is given by pos_{col}. We discretize the Y position as a fixed number of rows in the viewpoint, $view_{row}$. We compute the weighted average of the cost for each row, j, to estimate pointing time:

$$T_{point} = \sum_{j=1}^{view_{row}} P_j T_{point_j} \quad (11)$$

The time of each row in the viewpoint is calculated by a regular Fitts' Law model:

$$T_{point_j} = a_f + b_f \log_2 \left(1 + \frac{\mathrm{d}((pos_{col}, j), view_{ctr})}{W}\right) \quad (12)$$

where d is the euclidean distance between a given target position and the viewport center, $view_{ctr}$. a_f ad b_f are to be learned. The probability for the target to be on each row j is determined by a probability density of normal distribution to reflect how the Y positions are distributed across the screen. μ and σ are empirically determined.

$$P_j = \frac{1}{\sigma\sqrt{2\pi}} exp(-(j/view_{row} - \mu)^2 \Big/ 2\sigma^2) \quad (13)$$

MODEL EVALUATION

To evaluate our model, we conducted a two-fold cross-validation based on the observed (study) and the predicted (model) data. In the following, we describe model implementation, parameter estimation, and evaluation results. Overall, we find parameters align with the observed data and the validation shows a high prediction quality.

Model and Training Implementation

To implement the model we use Python based Tensorflow [1] as a status-quo machine learning library that provides an easy and efficient method to estimate parameters for a model that involves multiple components.

Implementation method. We implemented the entire model as a whole, including data handling, parameter learning, and each formula directly implemented as a set of Tensorflow operations in Python. For further details on implementation, we refer to the corresponding API.

Training algorithm. We employ the Stochastic Gradient Descent (SGD) optimizer in TensorFlow for model training and parameter estimation. SGD is a common optimization algorithm for parameter estimation. Specifically, we used tf.train.AdagradOptimizer in the Tensorflow API with learning rate 0.1 and batch size 32 for 100000 iterations, at which learning reached a plateau.

Parameter clipping. We use clipping (tf.clip_by_value) as a method to provide constraints to input parameters. Particularly we enforce certain parameters such as the intercept of navigation (Eq. 3/4) or both parameters of Fitts Law (Eq. 12), to remain positive because they represent added time cost only meaningful when positive.

Split training. Split training has been used to, e.g., to calibrate motor performance independent of other factors [7, 23]. As initial tests showed our model had a bias toward the top-down navigation strategy as it occurred more frequently in the training data (80%, Fig. 5), we employ split training to first train strategy factor S_{prob} (Eq. 7), and then train the entire model with the given parameters. To train strategy, we generated a dataset where each strategy occurs equally (i.e., adding bottom-up data points randomly selected from the existing dataset until reaching equal amount of samples).

Parameter Estimation

For strategy S_{prob}, the learned weights, s_i, are [-3.067, -1.701, -0.414, 19.257]. The large magnitude of weight for l, 19.257, indicates that the initial letter of a target name has a strong effect on which strategy the user chooses to use, while the rest factors are negligible. Navigation cost intercepts were b_{tdn}=0.6 and b_{bup}=1.44, showing that the bottom-up navigation starts with the a higher offset that fits to the initial flick-scroll down gesture that users perform to get to the bottom. Lastly, the cost for navigating each row results in $T_{row} = 0.1\exp(-0.277t) + 0.096$.

There is a minor effect on columns due to the rapid visual search: $T_{col} = 0.606\exp(-0.805t) + 0.0002$, and $v = 0.05$ for T_{vs}. For Fitts' law, a_f = 0.025 and b_f = 0.001, indicating this component did not have a significant impact on the time performance either. We speculate our training process was not able to pick up marginal time differences caused by visual search and pointing components, which are far smaller than the cost of the navigation sub task in a grid task.

Model Accuracy

We conducted a 2-fold cross-validation on the study datasest, in which the model is trained with a half of the users (10 users), and tested on the other half. We report on the accuracy of our model in predicting time performance for grid UI as R^2, which measures the correlation between the observed time performance and predicted one. The average is taken from all samples within one condition. The particular conditions are reported next to the R^2 results. Figure 12 shows comparative results on observed and predicted times.

The model predicted the user's performance across blocks with an accuracy of $R^2 = .99$ (8 blocks). Figure 12a shows how the power law of practice applies well to our grid interaction.

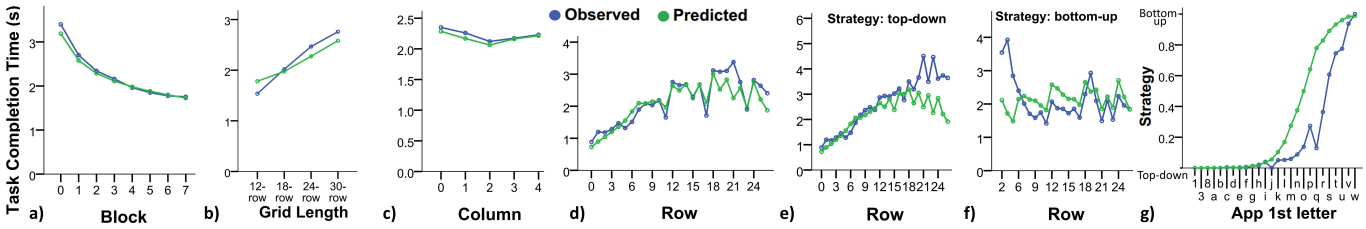

Figure 12. Observed and predicted menu selection time across various factors of grid performance.

Another main factor, grid length, has also been well modelled at $R^2 = .97$ (8 block×4 grid), as indicated in Figure 12b.

We also analyzed the model fit for factors column and grid. Column is integrated in the visual search equation, and resulted in an accuracy of $R^2 = .86$ (5 col×4 grid). As visible in Figure 12c, it is accurately modeled how users are faster in the central column region.

The row defines vertical target position, thus is integral in capturing navigation performance. Our model is able to predict performance variance across rows (12, 18, 24 and 30 rows) with $R^2 = .83$, and Figure 12d shows fitting results of predicted time across rows. Training of strategy was slightly offset for the transition from top-down to bottom-up strategy; the model predicted users would use the bottom-up approach at earlier letters in the alphabet (Fig. 12g). One potential reason for this issue is that we did not control factor letter in the study, leading to some letters with higher, and others with lower frequency. Nonetheless for each strategy separated, we can see that our model still approximated the times well (Fig. 12e-f).

UTILITY-BASED ADAPTIVE UI

We discuss how the predictive model we developed can be applied in an adaptive user interface, which is aimed to reduce user effort by adapting the interface from predictions of next user actions. For example, in the App Launcher on Google Pixel devices, five suggestions of next apps to use are presented at the top of the screen based on the user's current situation. If a target app is among these predictions, the user can immediately launch it without searching the whole grid.

Probability-Based Optimization

A common way for deciding what items to suggest at a given step, t, is based on the probability distribution over all possible items, P_t, which are determined by an event prediction engine that is out of the scope of this paper. The event prediction engine may score each item (representing an action) based on a range of external signals such as time of the day or user location. Note that event prediction is different from time performance prediction that our model is designed for.

A probability-based method selects a given number of items that have the highest probabilities, e.g., 5, and placed them at a convenient location, e.g., the top of the screen (referred as the *prediction bar*), for easy access by the user. We formulate the time cost for accessing item i at trial t as the following.

$$cost_t^i = \begin{cases} C & \text{if } i \in Top5(P_t) \\ G(i,t,g) & \text{otherwise} \end{cases} \quad (14)$$

where $Top5$ picks 5 items that have the highest probabilities, and $G(i,t,g)$ is our model that predicts time performance for accessing item i given the trial, t, and with a given grid configuration, g. C represents a constant time for accessing an item in the prediction bar.

Utility-Based Optimization

We hypothesize that by considering the time cost for accessing each item in a predictive grid interface, in addition to the probability of the item, we can better reduce the user effort. We define the utility for suggesting an item as the product of its probability and its cost. We compute the utility for each item in the grid as the following.

$$U_t = P_t \odot G(t,g) \quad (15)$$

$G(t,g)$ computes the cost for each item in the grid based on the time performance model derived in this paper, which results in a vector. $\odot$ represents the pairwise product between the probability distribution and the cost vector. We formulate a utility-based optimization method as the following.

$$cost_t^i = \begin{cases} C & \text{if } i \in Top5(U_t) \\ G(i,t,g) & \text{otherwise} \end{cases} \quad (16)$$

The equation is similar to Eqn 14. except that $Top5$ selects items based on utilities instead of probabilities.

Simulation Experiments

To validate our hypothesis, we compare the two methods with task sequences generated based on two target distributions, i.e., the uniform and the Zipf [41], which both have been used in previous work for task distributions [19]. In a Zipf distribution, a small number of items are highly frequently used while the rest in the grid are rarely used. At each trial, we draw a probability distribution over all possible items from a Dirichlet that is seeded with one of these target distributions. We then draw the target for the trial by sampling the probability distribution. This approach has several benefits. First, the probability distribution is valid because the target is drawn from it. Second, the resulted task sequence is mostly consistent with the desired target distribution. Last, Dirichlet can be parametrized to simulate an item predictor with different accuracy, i.e., how often the item that has the largest probability is indeed the target.

We generated time sequences for completing 100 trials based on the 2 target distributions, 4 grid lengths and the two optimization methods. We also deliberately vary the item prediction accuracy from 10% to 70%. This simulation results in 1600 sequences, each with 100 trials. Based on this simulation, we compute the total time needed for completing a sequence

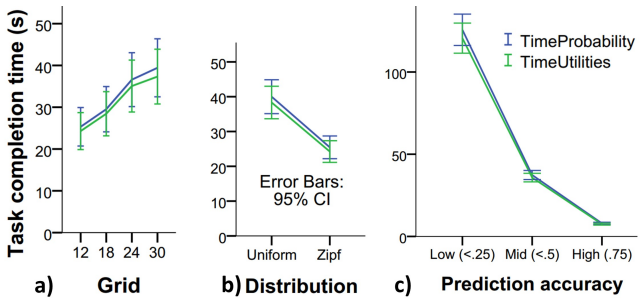

Figure 13. Simulation of utility- versus probability-based optimization for factor grid, distribution, item prediction accuracy. Across factors, the inclusion of utility benefits the performance by reducing task times.

using a given optimization method, grid length, target distribution as well as item prediction accuracy. Figure 13a-c show the results of the simulation.

We found the utility-based optimization outperformed the probability-based method in reducing task completion time (F_7^1=35, p<.0006; note that the large CI bars do not necessarily indicate statistical non-significance). Across all conditions, the utility method performs faster than the probability based method (Figure 13). When the grid has more items, the advantage of the utility-based method is more pronounced (a), considering that targets that are further down the menu will acquire higher utility to be placed at the prediction bar. We binned item prediction accuracy to three equal bins (low, mid, high accuracy). Here we also find that when the accuracy of item prediction is low, the gain of the utility-based method over the probability one is more pronounced (c).

DISCUSSION & LIMITATIONS

This work investigates interaction behaviors with a scrollable grid UI, a common task that people perform on mobile phones, showing many factors affect this simple interaction task. There are structural factors such as target position, row and column dimensions, and viewport size. There are gestural factors such as gesture amount, timing, speed or duration. Navigational factors play an important role such as strategies influenced by the target's name, selection with or without scrolling, and rapid flick-based versus continuous scrolling gestures. Our investigation covered a good range of these factors that exist in this common interaction behavior. More research is needed for investigating the rich space of mobile interaction.

One novelty of our model is the probabilistic estimation of navigation strategy to capture the switching between top-down and bottom-up navigation. Prior models such as SDP have been extended with scrolling navigation [17], however do not explicitly cover the mobile space or navigation strategies. It is an open question whether the complexity of menus or grids can be effectively modeled using a single approach, or whether each instance of these UIs in its own task environment needs to be modeled specially. Our work shows that there are many peculiarities to be considered for a rather basic selection task.

A limitation of our approach is that our model is informed by a controlled study. In a realistic application, Zipf [41] is a more appropriate distribution for menu or grid selection tasks. In addition, our study tasks are limited to interaction in a short period and with the goal to be as fast and accurate as possible. Ideally, the model should consider interaction over an extended period of time as well as other user preferences such as comfort. It is important to study how the user performance evolves in a realistic setting and how a performance model should be reformulated to account for the differences.

Another limitation lies in the model training and evaluation. While our model provides good accuracy in modeling human performance in grid interaction, we cannot sufficiently interpret each learned parameter value and validate these values by corresponding them to specific human cognitive and motor control behaviors. There are two aspects of the problem. One is that we need to better regulate the parameter estimation during the learning process such that the learned parameters are more sound based on existing theories. The other aspect is to better segment and label each task component to enable better guidance and validation for parameter estimation. However, this is challenging due to the interplay among components, which deserves further investigation.

In future we intend to model other grid UIs such as horizontal grids, which might involve different navigation and visual search patterns. Scrolling that allows 2 DOF such as panning a map may be modeled by integrating both the horizontal and the vertical navigation. In contrast to scrolling, discrete swipe is often used in horizontal grids to flip through pages, which can be modeled by adding a single swipe constant between grid pages. Integration of performance components such as user error, e.g., when selecting a wrong item resulting in additional correction time [8], would further complete the model.

We have theoretically discussed the novel idea of extending predictive user interfaces by considering the utility of each item, i.e. not only how likely an item is the target but also how costly (time) it is for the user to select the item. This improves interaction efficiency by integrating cost (as predicted by our model) into prediction of next user actions, which we will further explore in the future.

CONCLUSION

We present an investigation into scrollable two dimensional grid UIs on touchscreen mobile devices. The study contributed a set of useful findings for understanding user behaviors on using a mobile grid UI. Based on these findings, we devised a novel predictive model for performance time where many unique characteristics of grid UIs converge into one formulation. We also discussed how such a predictive model can be used for utility-based optimization of user interfaces. The work contributes to HCI research of user performance modeling, taking steps toward modeling the complex behaviors inherent in mobile user interaction.

REFERENCES

1. Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and

Xiaoqiang Zheng. 2016. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 265–283. https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf

2. Johnny Accot and Shumin Zhai. 1997. Beyond Fitts' Law: Models for Trajectory-based HCI Tasks. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '97)*. ACM, New York, NY, USA, 295–302. DOI: http://dx.doi.org/10.1145/258549.258760
3. David Ahlström, Andy Cockburn, Carl Gutwin, and Pourang Irani. 2010. Why It's Quick to Be Square: Modelling New and Existing Hierarchical Menu Designs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 1371–1380. DOI: http://dx.doi.org/10.1145/1753326.1753534
4. Tue Haste Andersen. 2005. A Simple Movement Time Model for Scrolling. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems (CHI EA '05)*. ACM, New York, NY, USA, 1180–1183. DOI: http://dx.doi.org/10.1145/1056808.1056871
5. John R. Anderson, Michael Matessa, and Christian Lebiere. 1997. ACT-R: A Theory of Higher Level Cognition and Its Relation to Visual Attention. *Hum.-Comput. Interact.* 12, 4 (Dec. 1997), 439–462. DOI:http://dx.doi.org/10.1207/s15327051hci1204_5
6. Gilles Bailly, Eric Lecolinet, and Laurence Nigay. 2016. Visual Menu Techniques. *ACM Comput. Surv.* 49, 4, Article 60 (Dec. 2016), 41 pages. DOI: http://dx.doi.org/10.1145/3002171
7. Gilles Bailly, Antti Oulasvirta, Duncan P. Brumby, and Andrew Howes. 2014. Model of Visual Search and Selection Time in Linear Menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 3865–3874. DOI: http://dx.doi.org/10.1145/2556288.2557093
8. Nikola Banovic, Tovi Grossman, and George Fitzmaurice. 2013. The Effect of Time-based Cost of Error in Target-directed Pointing Tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1373–1382. DOI: http://dx.doi.org/10.1145/2470654.2466181
9. Xiaojun Bi, Yang Li, and Shumin Zhai. 2013. FFitts Law: Modeling Finger Touch with Fitts' Law. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1363–1372. DOI: http://dx.doi.org/10.1145/2470654.2466180
10. Hans-Joachim Bieg, Lewis L. Chuang, Roland W. Fleming, Harald Reiterer, and Heinrich H. Bülthoff. 2010. Eye and Pointer Coordination in Search and Selection Tasks. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications (ETRA '10)*. ACM, New York, NY, USA, 89–92. DOI: http://dx.doi.org/10.1145/1743666.1743688
11. Matthias Böhmer and Antonio Krüger. 2013. A Study on Icon Arrangement by Smartphone Users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2137–2146. DOI: http://dx.doi.org/10.1145/2470654.2481294
12. Micheal D. Byrne. 2001. ACT-R/PM and Menu Selection. *Int. J. Hum.-Comput. Stud.* 55, 1 (July 2001), 41–84. DOI: http://dx.doi.org/10.1006/ijhc.2001.0469
13. Michael D. Byrne, John R. Anderson, Scott Douglass, and Michael Matessa. 1999. Eye Tracking the Visual Search of Click-down Menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. ACM, New York, NY, USA, 402–409. DOI:http://dx.doi.org/10.1145/302979.303118
14. Xiang Cao, Jacky Jie Li, and Ravin Balakrishnan. 2008. Peephole Pointing: Modeling Acquisition of Dynamically Revealed Targets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 1699–1708. DOI: http://dx.doi.org/10.1145/1357054.1357320
15. Stuart K. Card. 1982. User Perceptual Mechanisms in the Search of Computer Command Menus. In *Proceedings of the 1982 Conference on Human Factors in Computing Systems (CHI '82)*. ACM, New York, NY, USA, 190–196. DOI:http://dx.doi.org/10.1145/800049.801779
16. Géry Casiez, Daniel Vogel, Qing Pan, and Christophe Chaillou. 2007. RubberEdge: Reducing Clutching by Combining Position and Rate Control with Elastic Feedback. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST '07)*. ACM, New York, NY, USA, 129–138. DOI: http://dx.doi.org/10.1145/1294211.1294234
17. Andy Cockburn and Carl Gutwin. 2009. A predictive model of human performance with scrolling and hierarchical lists. *Human–Computer Interaction* 24, 3 (2009), 273–314.
18. Andy Cockburn and Carl Gutwin. 2010. A Model of Novice and Expert Navigation Performance in Constrained-input Interfaces. *ACM Trans. Comput.-Hum. Interact.* 17, 3, Article 13 (July 2010), 38 pages. DOI: http://dx.doi.org/10.1145/1806923.1806927
19. Andy Cockburn, Carl Gutwin, and Saul Greenberg. 2007. A Predictive Model of Menu Performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 627–636. DOI: http://dx.doi.org/10.1145/1240624.1240723
20. Michael D Fleetwood and Michael D Byrne. 2002. Modeling Icon Search in ACT-R/PM. *Cogn. Syst. Res.* 3, 1 (March 2002), 25–33. DOI: http://dx.doi.org/10.1016/S1389-0417(01)00041-9

21. William E Hick. 1952. On the rate of gain of information. *Quarterly Journal of Experimental Psychology* 4, 1 (1952), 11–26.

22. Ken Hinckley, Edward Cutrell, Steve Bathiche, and Tim Muss. 2002. Quantitative Analysis of Scrolling Techniques. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '02)*. ACM, New York, NY, USA, 65–72. DOI: http://dx.doi.org/10.1145/503376.503389

23. Jussi P. P. Jokinen, Sayan Sarcar, Antti Oulasvirta, Chaklam Silpasuwanchai, Zhenxin Wang, and Xiangshi Ren. 2017. Modelling Learning of New Keyboard Layouts. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 4203–4215. DOI: http://dx.doi.org/10.1145/3025453.3025580

24. David E. Kieras and David E. Meyer. 1997. An Overview of the EPIC Architecture for Cognition and Performance with Application to Human-computer Interaction. *Hum.-Comput. Interact.* 12, 4 (Dec. 1997), 391–438. DOI:http://dx.doi.org/10.1207/s15327051hci1204_4

25. David M. Lane, H. Albert Napier, Richard R. Batsell, and John L. Naman. 1993. Predicting the Skilled Use of Hierarchical Menus with the Keystroke-level Model. *Hum.-Comput. Interact.* 8, 2 (June 1993), 185–192. DOI: http://dx.doi.org/10.1207/s15327051hci0802_4

26. I Scott MacKenzie. 1992. Fitts' law as a research and design tool in human-computer interaction. *Human-computer interaction* 7, 1 (1992), 91–139.

27. James E McDonald, Jim D Stone, and Linda S Liebelt. 1983. Searching for items in menus: The effects of organization and type of target. In *Proceedings of the Human Factors Society Annual Meeting*, Vol. 27. SAGE Publications Sage CA: Los Angeles, CA, 834–837.

28. Brad Mehlenbacher, Thomas M. Duffy, and James Palmer. 1989. Finding Information on a Menu: Linking Menu Organization to the User's Goals. *Hum.-Comput. Interact.* 4, 3 (Sept. 1989), 231–251. DOI: http://dx.doi.org/10.1207/s15327051hci0403_3

29. A. Newell and P. S. Rosenbloom. 1993. The Soar Papers (Vol. 1). MIT Press, Cambridge, MA, USA, Chapter Mechanisms of Skill Acquisition and the Law of Practice, 81–135. http://dl.acm.org/citation.cfm?id=162580.162586

30. Erik Lloyd Nilsen. 1991. *Perceptual-motor Control in Human-computer Interaction*. Ph.D. Dissertation. Ann Arbor, MI, USA. UMI Order No. GAX91-35659.

31. Ken Pfeuffer, Jason Alexander, and Hans Gellersen. 2016. Partially-indirect Bimanual Input with Gaze, Pen, and Touch for Pan, Zoom, and Ink Interaction. In *Proc. CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, USA, 2845–2856. DOI: http://dx.doi.org/10.1145/2858036.2858201

32. Philip Quinn, Sylvain Malacria, and Andy Cockburn. 2013. Touch Scrolling Transfer Functions. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. ACM, New York, NY, USA, 61–70. DOI: http://dx.doi.org/10.1145/2501988.2501995

33. Michael Rohs and Antti Oulasvirta. 2008. Target Acquisition with Camera Phones when Used As Magic Lenses. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 1409–1418. DOI: http://dx.doi.org/10.1145/1357054.1357275

34. Joey Scarr, Andy Cockburn, Carl Gutwin, and Andrea Bunt. 2012. Improving Command Selection with CommandMaps. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 257–266. DOI: http://dx.doi.org/10.1145/2207676.2207713

35. Joey Scarr, Andy Cockburn, Carl Gutwin, and Sylvain Malacria. 2013. Testing the Robustness and Performance of Spatially Consistent Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 3139–3148. DOI: http://dx.doi.org/10.1145/2470654.2466430

36. Barton A. Smith, Janet Ho, Wendy Ark, and Shumin Zhai. 2000. Hand Eye Coordination Patterns in Target Selection. In *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications (ETRA '00)*. ACM, New York, NY, USA, 117–122. DOI: http://dx.doi.org/10.1145/355017.355041

37. Benjamin L. Somberg. 1987. A Comparison of Rule-based and Positionally Constant Arrangements of Computer Menu Items. In *Proceedings of the SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface (CHI '87)*. ACM, New York, NY, USA, 255–260. DOI: http://dx.doi.org/10.1145/29933.275639

38. The Statistics Portal Statistica. 2016. Statistica, The Statistics Portal. (2016). Retrieved Febuary 2, 2015 from https://www.statista.com/statistics/681206/us-app-installation-usage-device/.

39. Shumin Zhai, Barton A. Smith, and Ted Selker. 1997. Improving Browsing Performance: A Study of Four Input Devices for Scrolling and Pointing Tasks. In *Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction (INTERACT '97)*. Chapman & Hall, Ltd., London, UK, UK, 286–293. http://dl.acm.org/citation.cfm?id=647403.723823

40. Jian Zhao, R William Soukoreff, Xiangshi Ren, and Ravin Balakrishnan. 2014. A model of scrolling on touch-sensitive displays. *International Journal of Human-Computer Studies* 72, 12 (2014), 805–821.

41. G.K. Zipf. 1949. *Human behavior and the principle of least effort: an introduction to human ecology*. Addison-Wesley Press. http://psycnet.apa.org/record/2005-10806-009