Supporting Visual Search and Spatial Memory in a Mobile Application Launcher

by

Scott Bateman, Carl Gutwin, Manasi Shah and Nathaniel Brewer

Technical Report TR18-240
October 10, 2018

Faculty of Computer Science
University of New Brunswick
Fredericton, N.B. E3B 5A3
Canada

Phone: (506) 453-4566
Fax: (506) 453-3566
E-mail: fcs@unb.ca
http://www.cs.unb.ca

Supporting Visual Search and Spatial Memory in a Mobile Application Launcher

Scott Bateman
University of New Brunswick
Fredericton, New Brunswick
scottb@unb.ca

Carl Gutwin
University of Saskatchewan
Saskatoon, Saskatchewan
scottb@unb.ca

Manasi Shah
Nathaniel Brewer
University of New Brunswick
Fredericton, New Brunswick
mshah1@unb.ca
nathaniel.brewer@unb.ca

ABSTRACT
App launchers—the interfaces used to navigate, find and open apps on mobile phones—were originally designed for a small set of apps that fit on a single screen. However, current app launchers, based on horizontally arranged screens and folders that hide apps, make it difficult to find apps at a glance or to remember their location. This work presents SpaceLaunch, an app launcher that couples support for rapid visual search with interactions and layouts that support the development of spatial memory. In two studies, participants worked with more than 200 apps presented with three alternative interfaces, we show that visual search is fast for novices, and that the transition to expertise is better supported by SpaceLaunch's flat hierarchy and zoom-based interactions. SpaceLaunch provides a novel interaction for an extremely common task, demonstrating that zoomable interfaces on smaller mobile screens are a promising direction for the design of fast and efficient interactions.

Author Keywords
Spatial interfaces; zooming; app launchers; mobile.

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI).

INTRODUCTION
The number of apps people have on their mobile phone is steadily increasing. This increase coincides with the amount of time they spend on their devices and the amount of storage they have for installing apps [1]. The basic model for app launchers – the interface used for locating and starting up applications – places apps in a grid of a fixed size (typically, 20-24 apps per screen). When the grid on a screen is full, apps are distributed across multiple pages.

While this layout strategy worked well for the original iPhone, when all available apps could be placed on a single screen, a recent survey found people typically have between 60 and 100 apps installed [1]. To display 100 apps with the current page-based layouts, the default launchers on iOS 10 or Android 8 would require at least five pages. This can make it cumbersome and inefficient to find a particular app, because it can be difficult to remember which page contains an app. While folders can be used to reduce the number of pages needed (by creating a shallow hierarchy), folders may hide app icons (either partially or completely), making it difficult to know what apps are in a folder.

People can opt to search if they have difficulty finding an app using navigation. However, navigation is still preferred because search is less efficient for several reasons: search can be slower requiring more keystrokes, people can forget the name of the app they are looking for, and people often prefer navigation over search because it requires less cognitive effort [40]. As a result, search is often a 'method of last resort' for graphical representations of information [8].

At least part of the problem is that even though apps are laid out in a spatial fashion, current app launchers can work against spatial memory (e.g., by failing to maintain spatial stability) and require interactions that are less efficient than they could be (e.g., searching through folders).

In this work, we propose a new model for app launchers (called *SpaceLaunch*). SpaceLaunch is based on the principles of supporting visual search and the development of spatial memory, in order to provide rapid access to content and a smoother transition to expert modes of interaction [8,9]. SpaceLaunch demonstrates a novel app launcher design by including a flat hierarchy of apps [36], providing a spatial overview [10], providing groupings that act as landmarks [43], and supporting rapid visual search [8]. We also provide a tap-to-zoom interaction that allows accurate and rapid selections, while allowing users to rapidly build expertise through supporting their spatial memory.

Through two studies, we demonstrate that SpaceLaunch allows interactions that are faster than traditional app launcher layouts (either page-based or folder-based) for novices, and that it better supports a transition to expertise.

Despite app launchers being a daily interaction for most people around the world, very little work has investigated how they can be better designed. Our work contributes new model for app launchers that allows target apps to be rapidly located when users are novice and allow them to transition to expertise more rapidly. Our findings provide further evidence for the use of spatial memory as a guiding principle in the design of interactive systems, and highlights several new directions for further research into visual search and spatial memory for improved interface design.

RELATED WORK

Hierarchical vs. Flat Organization of Commands

A common method for organizing content in graphical user interfaces is to arrange items into a hierarchy – such as multi-level menus or the hierarchical toolbars of Microsoft's Ribbon. This allows items to be grouped in categories (e.g., the standard File, Edit, and View menus), which can improve novices' ability to find items [36]. However, there are two problems with hierarchical navigation.

First, once users are familiar with command locations, hierarchies impose an additional cost on execution because the user must still navigate through the hierarchy to get to the command. Several prior projects have proposed flattening command hierarchies as a way to reduce the added cost for experts (e.g., [16,34,36]). For example, the ListMap system [16] presented 225 fonts in a single visible grid, and showed that selection for experts was faster than a standard listbox presentation; similarly, the CommandMap interface [36] flattened the command hierarchies for Microsoft Word, and showed performance improvements over the Ribbon.

Second, hierarchies that are built by the user (such as folder hierarchies) may not be an optimal representation of semantics for purposes of finding and retrieval [21,24,25]. People may choose inappropriate categories (e.g., that are semantically ambiguous, such as a "Stuff" folder), often fail to realize that information may be needed in the future, and do not keep hierarchies and contents up to date [21,24]. In addition, items often have multiple semantics (e.g., should the Facebook app be filed into "Communication" or "Social"?) or semantics that change over time (e.g., from a "Current Tasks" group to a content-based group). Malone's early study of personal organizational schemes demonstrates how often these groupings can be misaligned: in two-thirds of the cases, documents were not filed under the category that the person used to describe them [24].

The problems associated with hierarchical structures (both misclassification and navigation depth) are a fundamental part of the study we describe below. One of our main principles in this work is that although flattened command structures can be intimidating for novices, they can prevent incorrect category selection, and reduce the number of actions required for expert users [36].

Spatial Memory as an Organizing Principle for UIs

Flattening a command hierarchy implies that all items are presented at the same level – leading to questions about how users will be able to find and remember specific commands. Previous research suggests that the main mechanisms are visual search (when users are inexperienced) and spatial memory (after users become familiar with the items) [9,36]. If a command's location is randomly chosen (and not based on some underlying principle such as alphabetic ordering), a novice will have to search the entire set for the desired command (with performance proportional to the linear number of items). Once the user is experienced, however, they can memorize the item's location, as long as the data remains spatially stable. Once learned, retrieval of the location is much faster than visual search – i.e., proportional to the base-2 log of the number of items memorized [8].

Several previous systems have used spatial organizations of information in order to flatten command hierarchies and improve expert performance. For example, the Data Mountain system [32] enabled fast memory-based retrieval of more than 100 web-page thumbnails (and the memories persisted over several months). Scarr and colleagues' CommandMaps system was tested both in a laboratory study and in a realistic field experiment, and showed that spatial memory is extremely effective once users have learned the locations of common items [34]. Similarly, the FastTap grid-menu technique [9] was developed for tablet interfaces, and a study of this system showed significant performance improvements both over standard menus and the well-known Marking Menu gesture technique [23].

Researchers have also looked at different aspects of how spatial memory is used in UIs, including work on how spatial transformations of the data (e.g., rotation or scaling) affects spatial retrieval [35], the role of effort in spatial learning [11], and the value of adding artificial landmarks in order to provide a stronger reference frame for spatial recall [43].

There is some evidence from work in desktop environments that suggest that spatial stability can support rapid app access. Tak et al. redesigned the desktop task switcher to provide a spatially stable layout that makes the most commonly accessed applications more salient, showing that it outperforms status quo task switching interfaces [39].

Zoom-based Interactions

When there are a large number of items in the dataset, a non-hierarchical presentation means that individual items may be too small for easy selection. This is particularly true on mobile touch devices where screens are smaller and where selection uses a finger rather than a mouse cursor. In these situations, changing the visual scale of the display may be necessary in order to allow visual confirmation and accurate touch targeting. Researchers have explored several types of zoom-based interaction to address these problems (e.g., [2,4,29]). The Pad system [29] and its successor Pad++ [4] were influential early investigations into the use of zooming as an alternative to hierarchical organization, and this work showed that if content layout is semantic (i.e., related things are close together), then zooming in on a group is similar to opening a hierarchical folder.

Zoom interactions have more recently been used as a navigation mechanism for large visual datasets such as photo collections [2,13] or document overviews [12]. For example, users of the PhotoMesa [2,22] and MediaFrame [13] systems visually search through a set of photo thumbnails, and then zoom in as a way to specialize and refine their search. In the Space-Filling Thumbnails system [12], page thumbnails are collected to form an overview of the entire document; users

can zoom in and out from the page view to the overview. Zoomboard applied a tap-to-zoom interaction to enable fast and accurate text entry on small smartwatch screens. An initial touch zooms into an area of the keyboard while the second tap, selects a key from the now larger subset [27]. In addition, researchers have looked at several types of zoom, including versions that maintain the context of the overview (e.g., fisheye lenses [37,38] or overview+detail views [30]), different kinds of animation between zoom levels (e.g., [3,41]), and zoom level that is dependent upon user actions such as scrolling speed [19].

Studies of Mobile Usage and Improving App Launching

Recent work has investigated how people organized apps and other functionality on their mobile phones [6,17,18]. This work has found people most frequently launch applications using app launchers as opposed to other means (including using notifications or from within other apps) [17]. Upon receiving a new mobile phone, customization of the app displays on the homescreen is one of the first activities that people do [18]; people arrange their icons by frequency of access, their relatedness to other apps, and even aesthetics [6, 17]. However, the most common motivation for customizing item placement is to make access as quick as possible [17].

Even though studies have suggested that people customize to support rapid app access, other researchers have recognized that organizing apps is not something that people like to do, and that many people do not take the time to do it [26]. In recognition of this tension, most work motivated by rapid access has focused on the related ideas of app recommendation [7,28], adaptive organization of apps [26], and adaptive placement [7,15] or visual highlighting [31]. These approaches are typically based on histories of access [28], contextual information (e.g., time of day) [5,28], and predictive models [28,42]. Very little work has shown that these adaptive approaches will work well in realistic usage, and that a potential limitation for such adaptive movement of apps is that people have a strong preference for easily understandable placement or rankings of apps [26].

Even though research clearly supports the idea that rapid access to apps is something that people would like, very little research has investigated how it can be supported through an improved mobile app launcher design.

CURRENT LAUNCHERS

Here we briefly describe current app launcher behaviour in iOS and Android, as they respectively account for 14.7% and 85% of the global market share in smartphones (Q1, 2017) [20]. Our description focuses on the use of general navigation mechanisms, and does not include search functionality and other interface widgets (e.g., weather forecasts) that are also common in modern launchers. It should be noted that Android devices often have custom launchers (either installed by the phone vendors, or installed from an app store by the user). We focus on versions of the app launchers that are present with the original version of Android 8 and iOS 10 (see Figure 1).

iOS App Launcher

iOS 10 displays up to twenty apps or folders in a 4x5 grid using a flow layout approach on each page. With iOS, all apps are placed in the launcher and can exist in only one place. Apps and folders are sized and labeled in the same way, occupying a specific location in the grid. When one page's grid is full, additional apps and folders are placed in the grid on a subsequent page (users can also manually move folders to other pages before the grid is full). The placement of pages is linear and stable. A small series of circles appear near the bottom of the page, where a page is represented by a single circle. To navigate, users can swipe (left or right) or touch the circles at the bottom of the page, which provides an animated transition advancing to the chosen page. A quick-launch bar with static content places four apps at the bottom of the page.

Folders can contain an unlimited number of apps (folders cannot be placed inside other folders). When the user opens an iOS folder, they see a folder page that shows the apps inside the folder – iOS folders can contain nine apps per folder page. Navigating to different pages within a folder can be achieved using the same circle and swipe metaphors as on the root-level pages. From the root level, folders display the first page of apps in miniature (i.e., up to nine apps). Selecting a folder from the main page switches to the folder page (with an animated zoom transition).

Figure 1. From left: iOS homescreen, iOS folder screen, Android home screen (enlarged to show a folder preview icon; top, center), and an Android folder screen.

Android App Launcher

The default launcher in Android 8 uses many of the same interactions as iOS. However, with Android, apps are not displayed in the main launcher by default. Instead an app drawer is provided with an alphabetical listing of apps. Apps can be selectively placed on the launcher page from the app drawer, where the basic iOS organization applies with a few minor differences. In Android, apps are organized in a 4x5 grid (on the home screen) or 5x5 grid (on subsequent pages). Items can be placed arbitrarily on the grid (i.e., not using a flow layout), and the quick launch bar at the bottom of the page contains 5 items (and can be dragged upwards to reveal the app drawer). Folders at the root level only show four items (i.e., all other items in folders are hidden). When folders are opened, a folder page shows a 4x4 grid of apps; if there are more than 16 apps in the folder, the behaviour of the folders is the same as iOS (including the use of circles to indicate and navigate pages).

Why Current App Launchers are Slow

Based on our review of the literature and description of how leading app launchers work, we identify three main problems that make finding apps slower and more cumbersome than necessary. Current app launchers:

1. Hide icons in folders which slows visual search and requires more interactions to perform search tasks;
2. Use page-based layouts which impose a deep hierarchy with a 1D arrangement of pages, requiring an increased number of interactions to navigate; and,
3. Increase the number of memory conflicts, because of the segmentation of the spaces (i.e., each page location can be occupied by multiple apps, but on different pages).

Hiding icons in folders and only showing a small subset of apps on the folder icons slows visual search, as users can not directly recognize a target unless it happens to be visible (see Figure 1). This causes problems for novices, who do not know the location of icons – to find an item, they must adopt a multi-step hunt and search strategy:

- The user must perform a visual search to see if the desired app is visible, and if so, select it.
- If the app is not visible, the user must reason about the potential location of an app within a folder, and consider whether they have missed the target during visual search.
- If a folder is opened, the user must return to step 1, continuing the visual search within the folder page.

Further, when apps are hidden in folders it is more difficult to build spatial memory, because a complete overview of the space is not available for viewing. This slows the ability to transition to more expert and rapid memory-based access.

Second, arranging apps in groups of pages imposes a deep hierarchy where pages must be accessed serially to navigate to a particular location. At each step through the hierarchy, a user may need to slow down to perform a visual search to look for the target icon and/or to orient themselves within the sequence of pages. This serial mode of access is slow and requires more interactions than even a folder-based access (where the 2D hierarchy allows direct, rather than serial, access to groups of apps). While some launchers do allow jumping ahead through the hierarchy (through a navigation bar), these typically lack semantic labels or visual cues, making recognition more difficult and error prone.

Third, the hierarchies of both pages and folders make it more difficult for users to learn spatial locations, because each location on the page is re-used multiple times. This means that valuable landmarks such as "the top left corner of the screen" become less useful, since the user must also maintain memory about which page or folder the landmark refers to.

SPACELAUNCH: A SPATIAL APP LAUNCHER

SpaceLaunch (see Figure 2) addresses the main shortcomings of current app launchers (as described above), by displaying a single flat, zoomable space for all apps. Our simple approach eliminates hierarchies and better supports rapid visual search and building spatial memory.

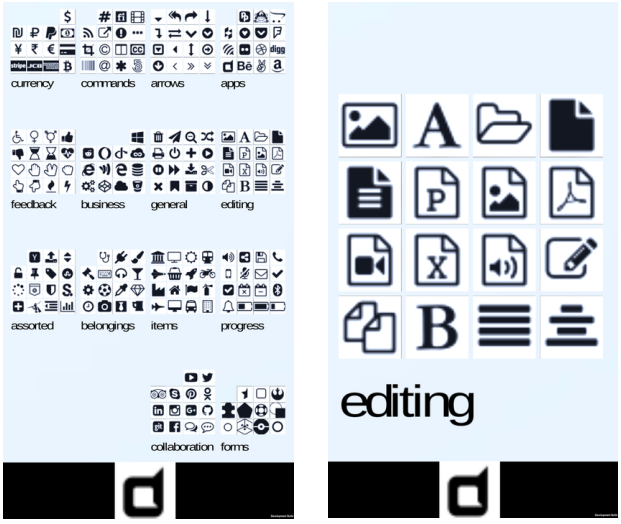

Figure 2. The main page of SpaceLaunch (left), and a zoomed in view of an app group (right).

To allow rapid navigation and simple interactions within the space, we adopt a two-stage selection process (like [27]). The main SpaceLaunch page initially provides and overview of the entire app space. When an area of the space is touched, a rapid, smooth transition zooms the view to a preset level centered around the point of the touch. Once zoomed, a second touch on an app icon selects the app icon. Returning to the zoomed out (overview) can be achieved through a pinch (or spreading fingers).

Further, our design maintains the spatial metaphor and allows exploration of the space while zoomed through panning. This allows fine adjustments to be rapidly made if a user misses an intended target or if they desire to view app icons at a larger size. Additionally, SpaceLaunch supports current conventions, by organizing apps into groups and providing semantic labels. Importantly, these groups do not hide any icons, and provide two main advantages. Groupings facilitate search for novices by providing semantics labels [33], and the groupings shapes and arrangement themselves act as landmarks that support spatial memory [43].

The size of the icons when zoomed out were carefully selected (based upon the size of app icons in folders on the iPhone 6s with iOS 10) to allow people with normal or corrected-to-normal vision to easily recognize the icons. The goal was to ensure that visual search is possible without the need to zoom in.

While there is much unused space in SpaceLaunch (as can be seen in Figure 2), we originally designed SpaceLaunch to accommodate a higher number of icons. By packing groups more closely together our design can comfortably display over 400 apps at the current icon size.

COMPARISON STUDIES OF SPACELAUNCH

We designed SpaceLaunch to address many of the shortcomings that exist with current app launchers. However, it was uncertain whether or not our design would lead to faster, easier and more efficient app selection when compared to status quo app launchers.

While our SpaceLaunch was informed by the design of several previous spatial interfaces, there were three main questions about our SpaceLaunch implementation that led us to question whether it would outperform current app launcher designs. First, SpaceLaunch uses small icons, and people need to visually search these icons quickly for our design to be effective. If the icons are too small or uncomfortable to use this would greatly slow search times. Second, with SpaceLaunch we will be showing over 200 icons at once (see Icon Sets, below), this substantial number of icons might be overwhelming and unusable because visual search and recognition of icons would be too difficult; 200 icons is towards the upper end of what has previously been explored for spatial interfaces. Finally, current app launcher designs that are built on app icons distributed across multiple pages and placed in folders, are already well entrenched interactions. It could be that the familiarity of these and expertise people have already developed with these interactions are too much to overcome.

Apparatus

We developed an experimental system using Unity 5.6 that provided 3 interfaces: SpaceLaunch (which worked as described above), a folder-based layout and a page-based layout (both described below). In place of the home bar, we placed a black bar that displayed the target icon that participants would search for during each trial. The experimental system was deployed on an LG G4 phone (5.5 inches; 2,560 x 1,440 pixels) for both studies.

Folder-based

We based our design of the folder-based layout on the way that Android 8 currently works (as described above; see Figure 3). In folder-based, all apps were placed in folders, which were arranged in a 4x4 grid. Folder icons, show only four apps. When a folder is touched all icons in the group are shown. Icons sets were purposively designed so that no more than 16 icons existed with the folders, to prevent the need for the use of multiple pages within folders. Once a folder was opened it could be closed using a pinch or spread gesture.

Page-based

The design of page-based was like the designs of Android 8 and iOS 10 (as described above; see Figure 3). App icons are arranged in a 4x4 grid on a series of pages laid out from left to right, and swipe motions changed pages. Page location indicator is shown as a series of circles below the apps, the highlighted circles indicate the current location in the extent of all pages. Individual circles could be touched to move to the corresponding page immediately.

Icon Sets

For study 1, we developed 3 icon sets for our experiment using the icons from Font Awesome (fontawesome.io/). We started by creating a set of black and white icons (to avoid visual pop-out effects caused by color) from all icons. We initially eliminated duplicates and icons that had little noticeable difference. This resulted in a set of 622 icons. We then grouped these icons into 14 groups that we judged to be most semantically similar. We then divided the larger set into 3 separate smaller sets of 207 icons in 14 groups each group having its own unique label, containing between 11 and 16 icons each. This meant that our 3 sets were distinct but were balanced in terms of their contents. These set were reviewed by the authors for consistency between groupings. Any unused icons were placed in our practice set.

It should be noted that we focused a great deal of effort on providing control and consistency for these icon sets; however, as described in our related work, there is no perfect grouping of items that would satisfy all uses cases and all individuals. We believe the variation of item placements that might still exist in our groups is likely a good rough approximation of what would exist in real world settings.

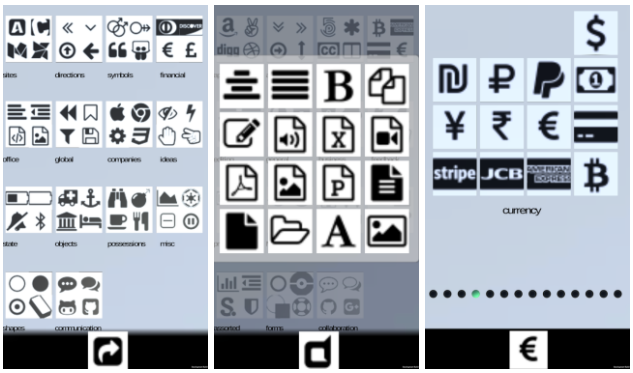

Figure 3. Images from the testing system. From left to right: the home page (in folder-based layout), an opened folder (in folder-based layout), and a single page (in page-based layout).

STUDY 1: COMPARISON WITH FOLDERS AND PAGES

In study 1, we were initially interested in whether SpaceLaunch could work at all and how it's performance might compare to existing launchers. We designed study 1 to determine if SpaceLaunch could be faster, easier and more efficient for accessing apps.

Participants

We recruited 9 participants (4 female) from a local university population. Average age 28.3 (sd: 12.6; min: 21; max: 61), 7 were students, 2 worked in full-time positions off-campus. Participants averaged 5.4 years of smartphone ownership. When asked how they organize and find app 6 participants used icons distributed across multiple pages (i.e., no folders) and 3 used a combination of folders and multiple pages. Only one participant used search frequently to find apps, 4 never use search, and 4 sometimes use search.

Procedure

Participants were explained the study procedure, and asked to complete an informed consent form and a short demographics questionnaire (focusing on their mobile usage). The study contained 3 interface conditions (page-based, folder-based and SpaceLaunch), whose presentation was fully counter-balanced with the three icon sets. Before starting the experimental tasks with each interface, the experimenter demonstrated each interface with the practice icon set, asking participants to practice until they were comfortable. Participants then started the experimental trials, finding 13 preselected icons in 10 blocks (130 trials per condition). The 13 stimulus icons were selected so that only two icons were in the same group and that only two icons would appear in the folder-based preview. The system paused after each block, so participants could take a break.

After each interface condition a questionnaire was given soliciting subjective judgments via a desktop computer. After the experiment, a final questionnaire was given asking the user to choose their most preferred interface. The experiment required approximately one hour to complete.

Data Collection

To assess if our system would be faster, easier and more efficient for accessing apps we collected three sets of metrics: for *speed*, we collected completion time of each trial; for *efficiency*, we collected the number of interactions required; and, for *ease*, we collected subjective measures including responses to the NASA TLX questionnaire, ratings of how "easy it was to remember app locations", and a forced selection question on the best interface for finding apps.

For each trial, our experimental system collected completion time and the touches needed to make a correct selection. The number of touches counts how many interactions a participant input up to and including the correct selection. Touches were dependent on the interface being used, but include selecting a folder, selecting an app, closing a folder, swiping to the next page, zooming-in, or zooming-out.

Subjective data was collected after each interface condition and after the experiment, using a 7-point Likert-scale-type questions. Participants also had opportunities to provide further insight into their judgments or other feedback via free-form text questions.

Data Analysis

Performance data were analyzed using a 3×10 RM-ANOVA, with interface (page-based, folder-based, SpaceLaunch) and block (1-10) as factors. Violations to sphericity used Greenhous-Geisser corrections to the degrees of freedom. Only significant pairwise differences are reported for post-hoc tests and only performed for interface conditions and not block, since we were less interested in performance differences between individual blocks. Post-hoc tests used Bonneferoni corrections. Subjective responses were analyzed using Friedman's test, and Likert-scale responses were recoded as numeric values (0-6, 3=neutral). Post hoc comparisons used the Wilcoxon signed-rank test. Free-form text answers are used to illustrate general trends in the findings. Before analysis, outlier trials were removed that were > 2 sd. away from the mean for completion time or touches, this resulted in the removal of 148 trials (4.2%).

Results of Study 1

Completion Time: There was a main effect of interface on completion time over all blocks ($F_{2,16}$=42.77, p<.001; see Figure 4). Pairwise comparisons showed that SpaceLaunch (mean=4.6s., σ=2.54) was significantly faster than both page-based (mean=8.00s., σ=3.49; p<.001) and folder-based (mean=5.58s., σ=3.92; p<.05) layouts, and folder-based was faster than page-based (p<.001). There was also a significant effect of block on completion time ($F_{9,72}$=59.20, p<.001). The interaction effect between interface and block was not significant ($F_{18,144}$=1.53, p>.05).

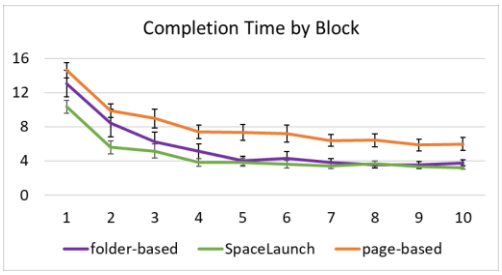

Figure 4. Mean completion time (±SE) by block in Study 1.

Touches: There was a main effect of interface on touches ($F_{2,16}$=51.15, p<.001; see Figure 5). Pairwise comparisons show that participants required significantly fewer touches to find target apps with SpaceLaunch (mean=2.20, σ=.14) than for folder-based (mean=9.85, σ=1.91; p<.005) and page-based (mean=13.17, σ=2.60; p<.001). Folder-based require significantly fewer touches than page-based (p<.001). There was also a significant effect of block on touches ($F_{9,72}$=59.20, p<.001). The small number of touches required to find apps with SpaceLaunch led to a significant interaction effect between block and interface on touches ($F_{18,144}$=4.64, p<.001).

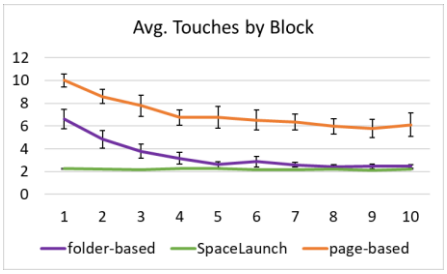

Figure 5. Mean touches(±SE) per selection by block in Study 1.

Subjective Data: When asked which of three systems was best for finding apps, participants overwhelmingly chose SpaceLaunch (8 chose SpaceLaunch, 1 chose folder-based, 0 chose page-based). When asked about their reasoning, participants made comments along the lines of "*… it was easier (sic) as you can see all the app[s]… [P9]*". However,

at least two participants commented that they preferred the folder-based approach because of its familiarity: "*Well, I felt most comfortable with the folders because I'm very familiar with them. However, the spatial felt like it had the most potential [P4]".*

There was significant difference between conditions for task loading, based on the NASA TLX ($\chi^2(2)=9.56$, $p<.01$; see Figure 6, left). Pairwise comparisons showed that SpaceLaunch had a significantly lower task loading than page-based ($z=-2.67$, $p<.001$), but there were no other differences. There was a significant difference between interfaces for 'ease of remembering app locations' ($\chi^2(2)=8.27$, $p<.05$; see Figure 6, right). Participants found it significantly easier to remember app locations with SpaceLaunch then page-based ($z=-2.37$, $p<.05$), but there were no other differences.

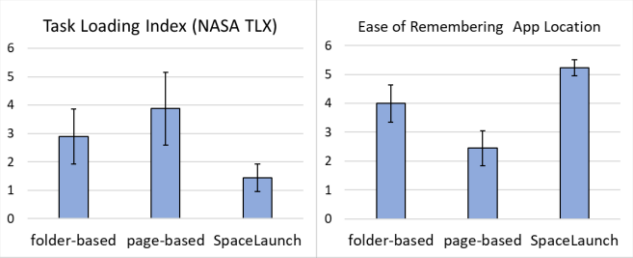

Figure 6. NASA TLX (left; lower is better), and ease to remember app locations (right; higher is better); from study 1.

Discussion of Study 1 Results

The results of study 1 are strongly in favor of SpaceLaunch. SpaceLaunch was fastest overall and most efficient. Participants also seemed to find SpaceLaunch the easiest and best interface for finding apps. The analysis provides convincing evidence that SpaceLaunch functioned as designed, and that the previously identified potential issues (small icon size, large number of icons, and expertise with alternatives) were not important factors.

It can be see that from Figure 4 and Figure 5 that participants made use of visual search effectively with SpaceLaunch. When users start out as novices, visually searching all app icons seems to be faster than searching smaller groupings of apps (which is required in folder-based and page-based), due to the additional interactions required. Although the app icons are relatively small in SpaceLaunch, they were still large enough for participants to successfully use them in visual search. If people needed to zoom, pan and inspect icons at larger sizes, the number of touch interactions would have been higher. However, the number of touch interactions (Figure 5) participants required was close to the minimum number of touches (a minimum of two touches are required to make a selection), right from block 1.

Visual inspection of Figure 4 seems to show a plateau for completion time around block 4 for completion time. This occurred when spatial memory comes into play and participants could rapidly recall the location of apps with a reduced need to perform slower visual search. This is evidenced by the performance floors in completion time shown in Figure 4, and further illustrated by the rapid decrease in the number of touches required by folder-based (see Figure 5). By block 5 folder-based achieves a similar plateau, indicating that participants were able to recall the location of apps within folders, even when the apps were hidden. However, the fact that SpaceLaunch could reach its performance floor earlier suggests that the spatial layout of SpaceLaunch supports a faster transition to expert behaviour.

STUDY 2: DIFFICULT LEARNING CONDITIONS

Our observations and conversations with participants in study 1, indicated that SpaceLaunch was clearly the fastest and most preferred method. However, the task in study 1 was a relatively simple learning task. Participants were aware that they were learning the location of a closed set of icons.

While SpaceLaunch outperformed folder-based and page-based layouts, we were interested in better understanding spatial learning and visual search performance in a more challenging task that better approximated real-world challenges. In real-world scenarios, people do not constantly rehearse access of a small set of apps; apps can be accessed infrequently, their interactions are separated by time and people's attention is divided among other tasks. These real-world conditions would cause issues including memory interference and memory decay (forgetting) might be particularly problematic for SpaceLaunch since it leverages spatial memory to facilitate the transition to expert, memory-based app selection.

To better approximate the difficulties for building memory in real world scenarios, we designed a second study, with only two interface conditions (SpaceLaunch and folder-based) that randomly presented distractors between stimuli. We opted to exclude page-based in Study 2, since it was the worst performing and the least preferred in study 1, and it would allow us to focus our study design on longer, more difficult tasks with the two leading interfaces.

Procedure

Study 2 used a similar procedure to study 1, but added in (0, 1 or 2) random distractor app icons between each of 10 stimuli (weighted such that the expected number of distractors was 5 per block), meaning on average participants saw 15 icons per block. We reduced the number of target icons to 10 per block to keep the total experiment time to one hour (determined through piloting).

By varying the number of distractors, we tried to ensure that participants were at least initially unsure which stimuli would be repeated and when they would be presented. Participants were not informed that there would be any repeated icons. Although, this is a small manipulation to our study 1 design, we found that it was extremely effective at increasing the difficulty of learning app locations (which we discuss more fully below).

Participants

We recruited 16 participants (4 female) from a local university population. Average age 23.3 (sd:4.0; min: 19; max: 37), 15 were graduate or undergraduate students (from a variety of disciplines), 1 worked in a full-time position off-campus. Participants averaged 6.0 years of smartphone ownership. When asked how they organize and access their apps, 8 participants use pages only, and 8 use a combination of folders and pages. Only two participant used search frequently to find apps, 7 never use search, and 7 sometimes use search.

Apparatus, Data Collection and Analysis

We collected the same data and followed the same analysis steps as in study 1, using a 2×10 RM-ANOVA, with interface (folder-based and SpaceLaunch) and block (1-10) as factors. However, we additionally split our analysis by whether a target app was *rehearsed* (an app icon that was part of the test set an was a repeated target 10 blocks) or *non-rehearsed* (was one of the distractor icons, selected at random from the icons outside of the test set). This allows us to better understand how learning progresses during realistic challenges to building memory. Before analysis, outlier trials that were > 2 sd. away from the mean for either completion time or touches were removed (227 trials = 4.3%).

Icon Set

Since there were only two interface conditions in study 2, we need only two unique icon sets for our experiment. However, we also increased the size of each of the two sets by creating an additional icon group of 11 app icons (15 groups in study 2, instead of 14 groups in study 1). We did this to increase the total number of distractors available and to increase the difficulty of searching and in building memory.

Study 2: Results

We present our performance results for study 2, split by whether the target was a rehearsed target (a stimulus, rehearsed each block) or a non-rehearsed target (a distractor, randomly presented). In this case, rehearsed targets is similar to the analysis done in study 1, while our design of study 2 allows us to additionally look at the non-rehearsed distractor targets.

Completion Time (Rehearsed Targets): There was a main effect of interface on completion time over all blocks ($F_{1,15}$=4.95, p<.05) for rehearsed targets (see Figure 7). SpaceLaunch (mean=5.5s., σ=3.74) was significantly faster than folder-based (mean=6.9s., σ=4.66). There was also a significant effect of block on completion time ($F_{2.95,44.25}$= 138.91, p<.001). The interaction effect between interface and block was not significant ($F_{2.94,44.13}$= 2.54, p>.05).

Touches (Rehearsed Targets): There was a main effect of interface on touches for rehearsed targets ($F_{1,15}$=25.79, p<.001; see Figure 8), SpaceLaunch (mean= 2.23, σ= 0.23) required fewer touches to find target apps than folder-based (mean= 4.63, σ=3.02). There was also a significant effect of block on touches ($F_{2.95,44.25}$= 39.73, p<.001). Again, the small number of touches to find apps with SpaceLaunch over all blocks led to a significant interaction effect between block and interface on touches ($F_{3.01,45.15}$=33.91, p<.001).

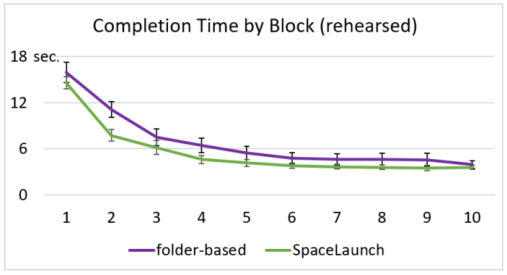

Figure 7. Rehearsed targets: mean completion time (±SE) by block in Study 2.

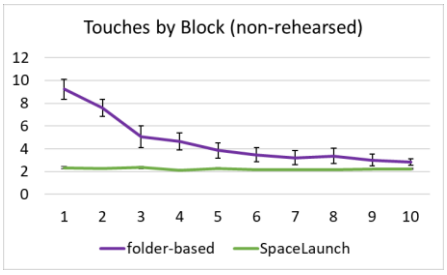

Figure 8. Rehearsed targets: mean touches (±SE) before a selection by block in Study 2.

Completion Time (Non-Rehearsed Targets): There was a main effect of interface on completion time over all blocks ($F_{1,10}$=13.4, p<.005; see Figure 9) for non-rehearsed targets. SpaceLaunch (mean=10.1s., σ=4.61) was significantly faster than folder-based (mean= 11.8s., σ=4.93). There was also a significant effect of block on completion time ($F_{4.26,42.57}$=2.90, p<.05). The interaction effect between interface and block was not significant ($F_{9,90}$= 0.62, p>.05).

While the general trend is downward for completion time (as shown in Figure 9), this must be interpreted in light of the fact that distractors can be presented as a target more than once (based on random chance). In practice, this means that by block 10, there was roughly a 15% chance that a distractor had previously been presented as a distractor earlier in the study. While this means that some 'rehearsal' was possible, with our non-rehearsal targets we believe this has a much smaller effect than building up expertise with the interfaces through repeated interactions, and incidental learning (see the discussion below).

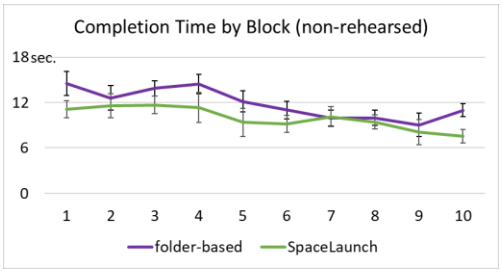

Figure 9. Non-rehearsed targets: mean completion time (±SE) by block in Study 2.

Touches (Non-Rehearsed Targets): There was a main effect of interface on touches for non-rehearsed targets ($F_{1,10}$=152.9, p<.001; see Figure 10), SpaceLaunch (mean=2.28, σ=.34) required fewer touches to find target apps than folder-based (mean=7.69, σ=3.59). There was no main effect of block on touches ($F_{4.78,32.53}$=2.26, p>.05). The small number of touches required to find apps with SpaceLaunch led to a significant interaction effect between block and interface ($F_{9,90}$=11.02, p<.05).

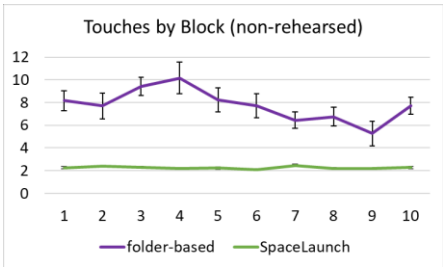

Figure 10. Non-rehearsed targets: mean touches (±SE) by block in Study 2.

Subjective Data: While SpaceLaunch continued to outperform folder-based in terms of our collected metrics, interestingly the subjective reports were more split. When asked which of three systems was best for finding apps, participants were split (9 chose SpaceLaunch, 7 chose folder-based). Like in study 1, participants who chose SpaceLaunch felt that the ability to visually search was a big plus: "*I can do a [visual] sweep of all the icons without having to 'open' folders [P7]*". Some participants simply felt remember locations of apps was easy and that "*... it seemed to come more naturally. [P5]*" However, participants who chose folder-based would often identify issues that did not come up in study 1: *"Spatial-based screen too busy (sic), too much stuff... [P16],"* and some did feel that the "*... icons [were] too small in spatial [P15].*"

Participant ratings of NASA TLX (z=-.958, p>.05) and "ease of remembering app location" (z=-1.54, p>.05) did not show significant differences (see Figure 11).

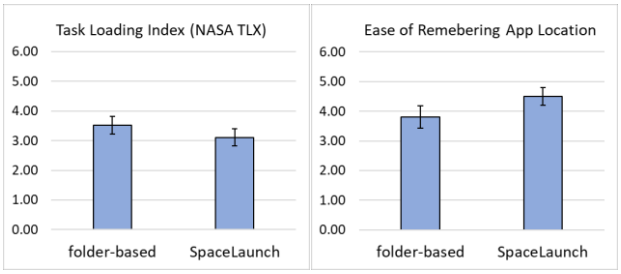

Figure 11. NASA TLX (left; lower is better), and ease to remember app locations (right; higher is better); from study 2.

Discussion of Study 2 Results

The simple addition of randomly inserted distractors (non-rehearsed) between repeated icons (rehearsed) resulted in a much more challenging task. Overall, SpaceLaunch was still the fastest and most efficient approach. This was true both when targets were rehearsed (Figure 4 and Figure 5) or when the targets were non-rehearsed (Figure 7). We discuss these two situations in turn below.

SpaceLaunch: Faster & More Efficient for Rehearsed Targets

Due to the increased difficulty of the experimental task, completion times show a more gradual learning curve for rehearsed targets. Figure 7 shows that the flattening of the learning curve that is reached by SpaceLaunch around block 8 is only reached for folder-based by block 10. The results for touches (see Figure 8) clearly show that participants were learning locations throughout the experiment, only approaching the minimum number of touch interactions by block 10 (two touches are required for folder-based selections). These results provide further evidence that SpaceLaunch better supports a quicker transition to expert memory-based retrieval, and that this transition can occur even under more challenging task conditions.

SpaceLaunch: Faster & More Efficient for Non-Rehearsed

SpaceLaunch is faster for non-rehearsed targets (see Figure 9), and is more efficient, with the number of touch interactions (see Figure 10) remaining consistently low. This means that even when participants were thrown back into a beginner mode of access, based primarily on visual search, SpaceLaunch was still fast and efficient. With folders, unrehearsed targets required much more searching through folders to find unknown targets, which explains the erratic curve for touches and higher variances throughout.

Why didn't people find SpaceLaunch easiest?

Interestingly, even though participant performance was clearly better using SpaceLaunch, participants seemed less aware of this advantage than in study 1. Opinions were split with regards to the ease of the approach. We often observed participants in study 2 stopping to ponder which approach was best at the end of the study. The reasons for the uncertainty towards SpaceLaunch are not entirely clear.

Participant reactions after completing a series of rehearsed targets and then a new non-rehearsed target suggested that switching between expert and beginner mode seemed to be a source of stress. Several participants seemed to be more comfortable with being active, and preferred to perform brute-force folder search over visual search (where they sat idle looking for the icon). These experiences may have create a paradoxical situation. With folders, participants always needed to be interacting with the system (whether a target's location was known or unknown). However, with SpaceLaunch the behaviour is very different for known targets (where a user interacts with the system) and for unknown targets (where a user must perform visual search). More study is needed to investigate this idea more fully.

Fatigue may have also played a role in study 2, where the length of the study and the higher number of repeated visual search tasks may have fatigued users. The smaller icons sizes of SpaceLaunch were not an issue for the shorter, easier tasks in study 1. However, they may have been too small for extended use. While fatigue did not show a detriment in

performance for SpaceLaunch, at least two participants did indicate that they felt the app icons were too small. We leave a fuller investigation of icon size and fatigue to future work.

OVERALL DISCUSSION

Our studies show that SpaceLaunch, with its zoomable interface and flat hierarchy, supports rapid visual search and a transition to expert memory-based app retrieval. In this discussion, we explore some remaining questions of generalization and research directions raised by the study.

Would people use SpaceLaunch?

To date there has been very little work exploring how app launching, an extremely common task, can be improved. We have articulated specific limitations with current app launchers, and shown how a novel design can provide substantial improvements in efficiency. However, there are also several existing app launchers that would allow an approximation of SpaceLaunch. For example, one could manually arrange up to 180 apps on the iOS homescreen by using the existing folder mechanism. The Apex Launcher for Android [44] allows the number of apps to be increased on the home screen by decreasing icon size. The Lens Launcher for Android [45] displays all apps on a single page in miniature, allowing zooming through a fish-eye lens interaction. These examples provide further evidence that there is interest in new app launcher designs. The fact that many people download custom app launchers and customize their mobile interfaces suggests that there is interest in improving the user experience, even with transient interactions such as app launching.

Would SpaceLaunch make a difference in real use?

While our experimental tasks are not highly realistic, we believe our second study approximates some of the difficulties that would be experienced under real world conditions. In actual usage, apps are accessed less frequently, and some degree of memory decay and memory interference would occur (which study 2's distractor targets also induced). Our future plans for SpaceLaunch include rebuilding it as a custom Android app launcher that we can deploy on the app store and collect data from use in the wild.

There are further questions that need to be explored in considering how our results generalize to real-world use. In particular, people make their own folders of apps, which would certainly play a role in anchoring app location in memory. Comparing user-created organizations with both pre-determined groups and a spatial interface like SpaceLaunch would be an interesting direction for future work. It is worth noting, however, that user control over app grouping is unlikely to solve the problem of poor organization: people are not particularly good at creating organizations for their apps, nor do they show much interest in doing it [24]. While our design of SpaceLaunch did incorporate labeled groups, this was largely to provide useful landmarks for navigation and to demonstrate that the design can accommodate a familiar organization scheme. Because spatial memory allows direct recall of item locations, it is likely that SpaceLaunch would work equally as well without any type of semantic grouping, and artificial landmarks [43] could be used to provide anchors for spatial memory.

For mobile phone owners who have relatively few apps, SpaceLaunch may provide little additional benefit. However, as discussed above, many users already have between 60 and 100 apps, and many have more [1]. As mobile storage continues to grow, people will have little reason to delete their apps. Mobile apps are an example of a growing information space, and SpaceLaunch shows another novel application of spatial interfaces that addresses a growing interaction need. It is also likely that SpaceLaunch could have other applications, for example, as a file browser or a way to manage browser windows or bookmarks.

Further Research on Spatial Interfaces

Past work has shown that for spatial memory to work, item locations need to be stable [8]. While this can be partially handled by affixing apps to a grid and not allowing them to move (unless initiated by a user), adding new apps and deleting old ones is relatively common. There is little work on how to maintain spatial memory as information spaces evolve, grow, and shrink, other than a few early studies [35]. This is a challenge for all spatial interfaces, and will be particularly important for SpaceLaunch.

Further, most research into spatial interfaces has focused on the use of visual search as the sole means of locating items. Current app launchers typically return search results as a simple ordered list, which does not support the acquisition of expert access behaviour (i.e., the app's location). We believe a solution like "search driven navigation" [14], where a search result highlights the path to the target app would assist users in learning app locations. We plan to incorporate such an idea into future versions of SpaceLaunch.

CONCLUSION

We present SpaceLaunch, a novel app launcher that supports the rapid finding of apps through visual search and the development of spatial memory. Through two studies, we demonstrate that SpaceLaunch is faster than traditional app launcher layouts (either page-based or folder-based) for novices, and that it better supports a transition to expertise. Further, our studies show that SpaceLaunch works in challenging tasks where building spatial memory can be more difficult. Our findings provide further evidence for the use of spatial memory as a guiding principle in the design of interactive systems, demonstrate that zoomable interfaces on smaller mobile screens are a promising direction for the design of fast and efficient interactions, and highlights several new areas for further research into visual search and spatial memory for improved interface design.

ACKNOWLEDGMENTS

Withheld for review.

REFERENCES

1. App Annie. *Spotlight on Consumer App Usage: Part 1.* Online report. Accessed on: July 14, 2017.Accessed at:

http://files.appannie.com.s3.amazonaws.com/reports/1705_Report_Consumer_App_Usage_EN.pdf

2. Benjamin B. Bederson. 2001. PhotoMesa: a zoomable image browser using quantum treemaps and bubblemaps. In Proceedings of the 14th annual ACM symposium on User interface software and technology (UIST '01). ACM, New York, NY, USA, 71-80. http://dx.doi.org/10.1145/502348.502359
3. Benjamin B. Bederson and Angela Boltman. 1999. Does Animation Help Users Build Mental Maps of Spatial Information?. In Proceedings of the 1999 IEEE Symposium on Information Visualization (INFOVIS '99). IEEE Computer Society, Washington, DC, USA, 28-.
4. Benjamin B. Bederson and James D. Hollan. 1999. Pad++: a zooming graphical interface for exploring alternate interface physics. In Readings in information visualization, Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA 530-543.
5. Matthias Böhmer and Gernot Bauer. 2010. Exploiting the icon arrangement on mobile devices as information source for context-awareness. In Proceedings of the 12th international conference on Human computer interaction with mobile devices and services (MobileHCI '10). ACM, New York, NY, USA, 195-198. DOI: https://doi.org/10.1145/1851600.1851633
6. 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: https://doi.org/10.1145/2470654.2481294
7. Matthias Böhmer and Antonio Krüger. 2014. A Case Study of Research through the App Store: Leveraging the System UI as a Playing Field for Improving the Design of Smartphone Launchers. Int. J. Mob. Hum. Comput. Interact. 6, 2 (April 2014), 32-45. http://dx.doi.org/10.4018/ijmhci.2014040103
8. Cockburn, A., Gutwin, C., Scarr, J., & Malacria, S. (2015). Supporting novice to expert transitions in user interfaces. ACM Computing Surveys (CSUR), 47(2), 31.
9. Gutwin, C., Cockburn, A., Scarr, J., Malacria, S., & Olson, S. C. (2014, April). Faster command selection on tablets with FastTap. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 2617-2626). ACM.
10. Andy Cockburn , Amy Karlson , Benjamin B. Bederson, A review of overview+detail, zooming, and focus+context interfaces, ACM Computing Surveys (CSUR), v.41 n.1, p.1-31, December 2008.
11. Cockburn, A., Kristensson, P. O., Alexander, J., & Zhai, S. (2007, April). Hard lessons: effort-inducing interfaces benefit spatial learning. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 1571-1580). ACM.
12. Cockburn, Andy, Carl Gutwin, and Jason Alexander. "Faster document navigation with space-filling thumbnails." Proceedings of the SIGCHI conference on Human Factors in computing systems. ACM, 2006.
13. Drucker, S. M., Wong, C., Roseway, A., Glenner, S., & De Mar, S. (2004, May). MediaBrowser: reclaiming the shoebox. In Proceedings of the working conference on Advanced visual interfaces (pp. 433-436). ACM.
14. Stephen Fitchett, Andy Cockburn, and Carl Gutwin. 2013. Improving navigation-based file retrieval. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13). ACM, New York, NY, USA, 2329-2338. https://doi.org/10.1145/2470654.2481323
15. Yusuke Fukazawa, Mirai Hara, Masashi Onogi, and Hidetoshi Ueno. 2009. Automatic mobile menu customization based on user operation history. In Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '09). ACM, New York, NY, USA, , Article 50 , 4 pages. http://dx.doi.org/10.1145/1613858.1613921
16. Gutwin, C., & Cockburn, A. (2006, May). Improving list revisitation with ListMaps. In Proceedings of the working conference on Advanced visual interfaces (pp. 396-403). ACM.
17. Alina Hang, Alexander De Luca, Jonas Hartmann, and Heinrich Hussmann. 2013. Oh app, where art thou?: on app launching habits of smartphone users. In Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services (MobileHCI '13). ACM, New York, NY, USA, 392-395. DOI: http://dx.doi.org/10.1145/2493190.2493219
18. Jonna Häkkilä and Craig Chatfield. 2006. Personal customisation of mobile phones: a case study. In Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles (NordiCHI '06), Anders Mørch, Konrad Morgan, Tone Bratteteig, Gautam Ghosh, and Dag Svanaes (Eds.). ACM, New York, NY, USA, 409-412. http://dx.doi.org/10.1145/1182475.1182524
19. Igarashi, Takeo, and Ken Hinckley. "Speed-dependent automatic zooming for browsing large documents." Proceedings of the 13th annual ACM symposium on User interface software and technology. ACM, 2000.
20. International Data Corporation. Smartphone OS Market Share, 2017 Q1. https://www.idc.com/promo/smartphone-market-

share/os. Accessed: 2017-09-10. Archived at http://www.webcitation.org/6tNOwBzMs

21. Jones, W., Phuwanartnurak, A. J., Gill, R., and Bruce, H. Don't take my folders away!: organizing personal information to get ghings done. In Ext. abstracts CHI '05, ACM (2005), 1505–1508.
22. Amir Khella and Benjamin B. Bederson. 2004. Pocket PhotoMesa: a Zoomable image browser for PDAs. In Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia (MUM '04). ACM, New York, NY, USA, 19-24. http://dx.doi.org/10.1145/1052380.1052384
23. Gordon Kurtenbach and William Buxton. 1993. The limits of expert performance using hierarchic marking menus. In Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems (CHI '93). ACM, New York, NY, USA, 482-487. http://dx.doi.org/10.1145/169059.169426
24. Thomas W. Malone. 1983. How do people organize their desks?: Implications for the design of office information systems. ACM Trans. Inf. Syst. 1, 1 (January 1983), 99-112. http://dx.doi.org/10.1145/357423.357430
25. Nardi, B., Anderson, K., and Erickson, T. Filing and finding computer files. Proc. EWHCI (1995).
26. Lauren Norrie and Roderick Murray-Smith. 2016. Investigating UI Displacements in an Adaptive Mobile Homescreen. Int. J. Mob. Hum. Comput. Interact. 8, 3 (July 2016), 1-17. DOI: http://dx.doi.org/10.4018/IJMHCI.2016070101.oa
27. Stephen Oney, Chris Harrison, Amy Ogan, and Jason Wiese. 2013. ZoomBoard: a diminutive qwerty soft keyboard using iterative zooming for ultra-small devices. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13). ACM, New York, NY, USA, 2799-2802. DOI: https://doi.org/10.1145/2470654.2481387
28. Abhinav Parate, Matthias Böhmer, David Chu, Deepak Ganesan, and Benjamin M. Marlin. 2013. Practical prediction and prefetch for faster access to applications on mobile phones. In Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing (UbiComp '13). ACM, New York, NY, USA, 275-284. http://dx.doi.org/10.1145/2493432.2493490
29. Ken Perlin and David Fox. 1993. Pad: an alternative approach to the computer interface. In Proceedings of the 20th annual conference on Computer graphics and interactive techniques (SIGGRAPH '93). ACM, New York, NY, USA, 57-64. http://dx.doi.org/10.1145/166117.166125
30. Emmanuel Pietriga, Caroline Appert, and Michel Beaudouin-Lafon. 2007. Pointing and beyond: an operationalization and preliminary evaluation of multi-scale searching. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07). ACM, New York, NY, USA, 1215-1224. DOI: https://doi.org/10.1145/1240624.1240808
31. Antoine Ponsard, Kamyar Ardekani, Kailun Zhang, Frederic Ren, Matei Negulescu, and Joanna McGrenere. 2015. Twist and pulse: ephemeral adaptation to improve icon selection on smartphones. In Proceedings of the 41st Graphics Interface Conference (GI '15). Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 219-222.
32. George Robertson, Mary Czerwinski, Kevin Larson, Daniel C. Robbins, David Thiel, and Maarten van Dantzich. 1998. Data mountain: using spatial memory for document management. In Proceedings of the 11th annual ACM symposium on User interface software and technology (UIST '98). ACM, New York, NY, USA, 153-162. http://dx.doi.org/10.1145/288392.288596
33. Joey Scarr, Andy Cockburn, and Carl Gutwin. 2013. Supporting and Exploiting Spatial Memory in User Interfaces. Now Publishers Inc., Hanover, MA, USA.
34. Joey Scarr, Andy Cockburn, Carl Gutwin, Andrea Bunt, and Jared E. Cechanowicz. 2014. The usability of CommandMaps in realistic tasks. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14). ACM, New York, NY, USA, 2241-2250. https://doi.org/10.1145/2556288.2556976
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. https://doi.org/10.1145/2470654.2466430
36. 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. http://dx.doi.org/10.1145/2207676.2207713
37. Doug Schaffer, Zhengping Zuo, Saul Greenberg, Lyn Bartram, John Dill, Shelli Dubs, and Mark Roseman. 1996. Navigating hierarchically clustered networks through fisheye and full-zoom methods. ACM Trans. Comput.-Hum. Interact. 3, 2 (June 1996), 162-188. http://dx.doi.org/10.1145/230562.230577
38. Manojit Sarkar and Marc H. Brown. 1992. Graphical fisheye views of graphs. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92), Penny Bauersfeld, John Bennett, and Gene

Lynch (Eds.). ACM, New York, NY, USA, 83-91. http://dx.doi.org/10.1145/142750.142763

39. Susanne Tak, Andy Cockburn, Keith Humm, David Ahlström, Carl Gutwin, and Joey Scarr. 2009. Improving Window Switching Interfaces. In Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part II (INTERACT '09), Tom Gross, Jan Gulliksen, Paula Kotzé, Lars Oestreicher, Philippe Palanque, Raquel Oliveira Prates, and Marco Winckler (Eds.). Springer-Verlag, Berlin, Heidelberg, 187-200. http://dx.doi.org/10.1007/978-3-642-03658-3_25
40. Jaime Teevan, Christine Alvarado, Mark S. Ackerman, and David R. Karger. 2004. The perfect search engine is not enough: a study of orienteering behavior in directed search. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04). ACM, New York, NY, USA, 415-422. http://dx.doi.org/10.1145/985692.985745
41. Barbara Tversky, Julie Bauer Morrison, and Mireille Betrancourt. 2002. Animation: can it facilitate?. Int. J. Hum.-Comput. Stud. 57, 4 (October 2002), 247-262. http://dx.doi.org/10.1006/ijhc.2002.1017
42. Chunhui Zhang, Xiang Ding, Guanling Chen, Ke Huang, Xiaoxiao Ma, and Bo Yan. 2013. Nihao: A Predictive Smartphone Application Launcher. In *Mobile Computing, Applications, and Services: 4th International Conference, MobiCASE 2012, Seattle, WA, USA, October 11-12, 2012. Revised Selected Papers*, David Uhler, Khanjan Mehta and Jennifer L. Wong (eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 294–313. Retrieved from https://doi.org/10.1007/978-3-642-36632-1_17
43. Uddin, M. S., Gutwin, C., & Cockburn, A. (2017, May). The Effects of Artificial Landmarks on Learning and Performance in Spatial-Memory Interfaces. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (pp. 3843-3855). ACM.
44. Apex Launcher: https://play.google.com/store/apps/details?id=com.anddoes.launcher&hl=en
45. Lens Launcher: https://play.google.com/store/apps/details?id=nickrout.lenslauncher&hl=en