Supporting Location Learning in a Spatial App Launcher

by

Manasi Shah

B.E Computer Science, Gujarat Technological University (2014)

A Report Submitted in Partial Fulfillment
of the Requirements for the Degree of

Masters of Computer Science by Report

in the Faculty of Computer Science

Supervisor: Scott Bateman, Ph. D., Faculty of Computer Science

Examining Board: Suprio Ray, Ph. D., Computer Science, Chair

Bradford Nickerson, Ph.D., Computer Science

This report is accepted by the

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

December 2018

ABSTRACT

For many day-to-day activities, we depend on smartphone apps. With an increasing number of apps available to use, managing and finding them can be frustrating and time consuming. Current app launchers work well with a small set of apps, but when the number of apps grows, the smartphone operating system provides multiple screens and folders that hide apps, which makes it difficult to find apps at a glance or to remember their location. Spatial interfaces have been proposed to deal with such problems, by leveraging peoples' spatial memory to make finding apps easy, making interactions rapid and effortless. However, spatial interfaces only work when the location of an app is known. In this work, we introduce a modified search functionality, which helps a user to learn the locations of apps through a spatial interface for launching mobile phone apps. In our study, we compare the performance of the search-to-learn interface as compared to the standard search-to-launch style search interfaces found in current app launchers. Our study demonstrates how the added investment to learn app locations can add up to a long-term reduction in time to find apps.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Dr. Scott Bateman, for his valuable guidance throughout my masters studies. I could not have imagined having a better mentor for my masters. It would never have been possible for me to take this work to completion without his constant support and direction. He always encouraged me to contribute a good piece of work. Besides my supervisor, I would like to thank my report committee and Dr. Patricia Evans for their insightful comments and encouragement on my work which enhanced the quality of my work. I am thankful to my fellow students at HCI lab who helped me during my work and provided feedback when required. I would also like to thank the members of the faculty for their support. I would like to thank my father, Bhagwati Kumar Shah and my mother Susheela Shah for their immeasurable love and care. They have always encouraged me to explore my potential and pursue my dreams. I am grateful to my husband, Pratik Jethaliya who has been an immense support and inspiration to give more and beyond my potential. Finally, a special thanks to my brother Pratik Shah for motivating and supporting me to set high goals and work had to achieve them. This accomplishment would not have been possible without them. Thank you.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Spatial memory, the memory of locations, plays an important role in human cognition. Frequent interactions with things or places allow us to remember the location and access it naturally and effortlessly. When used in human computer interaction, spatial memory can have benefits to memory similar to those found in the real world. Learning the spatial location of controls in any interface makes interaction natural and rapid, without the need to perform a slow visual search. In order to utilize spatial memory, the item locations on the interface should be consistent [12]. However, many interfaces break a user's spatial memory by rearranging or segmenting items (e.g., by hiding them in folders), which becomes an obstacle to spatial learning.

As mobile phones have become an essential part of our daily lives, the number of apps people use is increasing. On an average, people have 60-100 apps in their phones [4]. Typical app launchers (the software used to find and select a phone application to use) segment app icons across multiple pages and would take at least 4-5 pages to store this number of apps. Finding a particular app can be a tedious task in current organization schemes because a user can only see the apps on the current page. Other icons are hidden from view, requiring several swipe interactions to reach a target page. Another option is to use folders, but folders do not display all the apps stored in them unless they are opened. Thus, both the pages and folder systems hide applications. These options are not suitable for developing spatial memory. People tend to use search options when they cannot find an app, but searching requires keystrokes, which requires additional effort

and increased interaction. Further, it is also possible that one could forget the name of the app he/she is looking for. In such cases, searching would be of no help. Hence, navigation is usually preferred over search [10], as it does not require the name of an app to be recalled. In this research we leverage a new design of a newly developed app launcher, called 'SpaceLaunch'. SpaceLaunch presents all apps on a single screen at the same level, where nothing is hidden from the user's view. This type of presentation makes all the items available for rapid visual search, and allows items to be spatially stable, which supports the development of spatial memory. This design makes the transition from a novice user to an expert user smoother [2], because once an app location is learned, visual search is no longer needed. SpaceLaunch was designed to fit hundreds of apps on a single screen. To allow this, the size of app icons is kept small, but still large enough to allow for visual search. Since the small size makes it difficult to select an item, we provide tap-to-zoom interaction that allows for accurate selections to be made. We also categorized apps into groups; these groups provide both spatial cues [14] for novice users, which act as landmarks that facilitate learning app locations, and as semantic organizations that allow inference about app location (i.e., Twitter will be in the "Social Media" area).

In our previous research, we found that SpaceLaunch, with its zoomable interface and flat hierarchy, supports rapid visual search and a transition to expert memory-based app retrieval [18]. While SpaceLaunch was faster when considering only navigation-based interactions, our previous work did not consider other means of finding and launching apps.

The fact that people use the search as a frequent means to find and launch apps cannot be neglected. However, current search interfaces for finding and launching apps do not utilize spatial memory. When search is used to find an app, the app launcher directly launches the app. This way the user does not know where the app is located, and they cannot develop memory of that location.

Since the search interface is the primary means to find an app when the location is not known, we also have developed a new method for searching, which encourages spatial learning. Our search-to-learn approach requires additional interactions to help the user to learn app locations and alleviates the need to search in the future. Upon initiating a search, the user gets a list of suggested apps based upon the key string they search for. On selecting an app from the list, the user is directed to the app location on the screen where the app is highlighted to stand out from other apps. Once an app is highlighted, the user can make a selection. Thus, searching for an app also helps in spatial learning.

In this work, we conduct a study to compare the performance of our search-to-learn approach for learning app locations in a spatial app launcher, with the status-quo search-to-launch interaction. Our study shows that with our search-to-learn method, searching for apps can become a learning process where people learn their app locations, which can lead faster interaction.

Our work is the first to demonstrate an important new search-to-learn interaction that will improve the viability of spatial interfaces in real world use. We demonstrate that while the required additional interactions of search-to-learn do slow down access time initially, these interactions translate into facilitated and faster access once app locations

have been learned. We also show that only four accesses to items are required to robustly learn the app locations. Our work is the first to demonstrate how common search interactions can be incorporated successfully into a spatial app launcher for mobile phones.

The remainder of this research report has the following organization. In Chapter 2 we provide a summary of related work that introduces command organizations, research into app launchers and how spatial memory is developed. In chapter 3, we compare the design and operation of existing IOS and Android app launchers and discuss their limitations. Then in chapter 4, we talk about the design of SpaceLaunch, its functionalities and how it overcomes the limitations of existing app launchers. In chapter 5, we describe our studies followed by the Results section (chapter 6), where we evaluate the data collected through our study and show that search-to-learn is faster than typical interfaces when a user has learnt the app locations. In chapter 7, we discuss how search-to-learn is demonstrated to be better than typical interfaces. Finally, we discuss future work and present several conclusions.

2. RELATED WORK

This study explores the effectiveness of spatial memory for graphical user interfaces, factors that affect spatial memory and its retrieval. Several areas of prior work are relevant to these objectives and are briefly reviewed in the following subsections.

2.1. Hierarchical vs. Flat Organization of Commands

In graphical user interfaces, commands are often organized in a hierarchical fashion where commands are categorized into different menus and submenus based on their behavior/usage. In general, this is done when there is a large number of commands available, so that novice users can narrow their search space by first determining the category of a command, and then by looking at a smaller number of commands. Microsoft's Ribbon is an example of such menus. However, there are two drawbacks for such hierarchical organizations.

First, users must navigate through the full hierarchy every time, even if they are aware of a command's location. This causes wasted physical effort and time to reach a desired command. To address this limitation, CommandMaps [17] used a flattened hierarchy and consistent spatial layout for accessing Microsoft Word commands. Through their study, Scarr, J. et. al [17] showed performance improvement in CommandMaps compared to the standard Ribbon menu. Similarly, ListMap [8] implemented a flat command hierarchy by presenting 225 fonts in a single grid, making all the fonts present at the same level and showed that it improved user performance compared to a standard listbox presentation.

Second, the users do not necessarily choose the right category, or they must think carefully before choosing the right category for a target item, which can lead to errors

and slowdowns. With SpaceLaunch we try to address problems with hierarchical organizations, where controls are hidden inside submenus and not reachable without going through entire hierarchy. This is also a limitation for current app launchers, which make use of hierarchies that hide app locations using pages and folders.

2.2. App Launchers

There have been some research focusing on how to design app launchers and the organization of apps faster for smartphones. Chunhui Zhang et. al [0] presented a predictive smartphone app launcher, 'Nihao', which predicts which app the user will likely open next based on a Bayesian Network model using contextual information such as the user's location, last used app and the current time. They found that this functionality did not necessarily affect app launching time. Instead, app launch time was highly dependent on the app icon position on screen. Thus, this study also favors users' spatial memory as a driving factor to find apps. However, Chunhui Zhang et. al [0] do not focus on utilizing spatial memory, despite it providing advantages for learning and recalling app icon locations. Our app launcher design supports spatial learning. Spatial learning makes the transition from novice user to an expert user smoother. It has been previously shown that casual or expert users take less time than novice users to navigate the interface [16].

2.3. Factors that affect spatial learning and retrieval

Several prior researchers provide information on factors that benefit spatial memory and should be considered while designing user interfaces. A *consistent spatial arrangement* of objects in the interactive interfaces provides several benefits. Spatial consistency allows users to develop memory of object locations, resulting in a reduced requirement

for visual search, which makes item recall and location rapid and effortless. Spatial memory has a large capacity [11] and it is long lasting [6]. Therefore, many research systems have used spatial memory as a basis for their interface design (e.g., CommandMaps [17]). Scarr et. al [12] performed a study to test user performance with a spatial interface that had been transformed with different degrees of translation, rotation, scaling, and perspective change. Results showed that users can quickly adapt to all forms of transformation but adapting to rotations was much slower than the other transformations. Scarr et. al [12] suggest that in mobile devices that allow landscape and portrait view modes, interface design should allow relative spatial consistency of items, rather than rearranging interface components to exploit the display space in the different layouts.

Another influencing factor is the use of spatial cues (or artificial landmarks). They are identifiable features in space that are easily discriminated from their surroundings [13], and which provide guidance for spatial actions in a familiar or unfamiliar environment. The default landmarks, like corners and borders of the screen, can be helpful for spatial learning. Uddin et al. [14] studied the effects of artificial landmarks on learning and performance. They carried out three studies to test the effects of three types of artificial landmarks: standard grid-lines, anchor points, and a semi-transparent image. They conclude that for small grid menus, default landmarks, like corners and edges of the screen, are sufficient to provide memory guidance; whereas for medium and large grids, the simple anchor marks significantly improved performance. This study used a small grid containing 64 items (8x8 grid) and large grid (10x16) containing 160 items. Simple visual anchors were found to be faster and less error-prone than visually rich,

transparent images [14], because an anchor menu provides row and column-based alignment, which image layout failed to offer. A study performed by Leifert [19] also concluded that visual grid lines help development of spatial memory. The experiment conducted for this study include two interfaces- one with visible grid (6x8) lines and one without them. For both interfaces, arrangement of objects on the screen had 2 variations: structured alignment and chaotic arrangement. Participants had to use their spatial ability to rearrange the objects on the board using cards with pictures of objects on them. The study concludes that grids are useful for supporting spatial memory, but they need to be carefully handled if content memory plays a role as well [19].

In many cases, a user's hand also plays a part in guiding spatial memory. A recent study [12] suggests the use of a user's own hands as landmarks. The result from this study showed that using the hand as a landmark significantly improved performance and allowed fast spatial memory development. The border or edges of the interface also acts as cue to remember the position of objects relative to the edges [12].

Using a third dimension in spatial interface design has long been a subject of discussion. Interactions with computers were re-encoded in terms of fidelity to the interactions with a real environment and consequently in terms of fitness to cognitive and spatial abilities. A further step in this direction was the creation of 3D displays, which amplified the fidelity of digital representations. However, there are no systematic results supporting the hypothesis that 3D displays better support cognitive spatial abilities [15].

A study to investigate implications of the third dimension on spatial memory was conducted by Tavanti and Lind [15]. The results found that 3D displays better supported

the task of correctly locating objects. However, researchers also hypothesized that perhaps a single visual property of any object (i.e., size) was used as a reminder for the memory task. Hence, this requires more research to identify if visual properties of the 3D space affected the performance.

Another study performed by Cockburn and McKenzie [1] suggested that a user's spatial memory degraded when using the 3rd dimension. They conducted an experiment to investigate the effectiveness of spatial memory in real-world physical models and in equivalent virtual systems. The interfaces were categorized in 2D, 2.5D, and 3D. The results show that the third dimension did not benefit user's spatial memory; in fact, it degraded the spatial memory by creating conflicts within 2D space [1].

In recent years, researchers have explored the use of animations in many aspects of user interfaces. Animation provides a continuous transition from one state of the interface to another. It has become increasingly common, both in research and commercial user interfaces. Bederson and Boltman [5] conducted a study to examine if animation helped users build mental maps of spatial information. They developed a system, Pad++. to explore ZUIs (zoomable interfaces). In their study, some users specifically stated that they thought the animations helped them learn the relationships between the data, others said that animations slowed them down. Animation had no significant effect on the task of navigating the family tree to learn the relationships between family members. For the task of reconstruction, animation had a significant effect. A study conducted by Donskoy, M. and Kaptelinin, V [7] showed that for a smooth animation, there must be several in-between frames shown quickly. However, their results do not precisely describe the effect of animation on navigation through the interface.

2.4. Spatial Memory Development and Retrieval

Discovering methods that help users to learn skills in using computer applications has long been an important research topic in HCI. Interface designers normally strive for a design that minimizes user effort. However, when the design's objective is to train users to interact with interfaces that are highly dependent on spatial properties, designers should consider explicitly increasing the mental effort of interaction [3]. While this might seem counter intuitive, Cockburn et al. [3] conducted a study to examine the benefits of effort-inducing interfaces on spatial learning. They designed a "frost-brushing" interface that forces the user to mentally retrieve spatial information. The results of a study showed that an effort-inducing interface improves spatial memory for both location and trajectory, when the amount of training time is experimentally controlled [3].

2.5. Search vs. Navigation

In any interface - mobile, desktop or tablet - search is often used when the item location is unknown, or it takes several steps (going through different folders or menus) to reach the item. Search seems to be a good option when a user is not sure where they should look for a particular item that they would like to access. However, Teevan et al. [9] found that participants in a study preferred navigating to known items even if they knew they could access the information most quickly through search. They refer to this behavior as an 'orienteering strategy', which allowed users to specify less of their information need and requires less cognitive effort.

3. CURRENT APP LAUNCHERS VS SPACELAUNCH

This section describes current app launcher behavior in iOS and Android, as they respectively account for 14.7% and 85% of the global market share in smartphones (Q1, 2017) [9]. We focus on versions of the app launchers that are present with the original version of Android 8 and iOS 10 (see Figure 1). Here we focus on the use of general navigation mechanisms and how the search functionality works in current app launchers. We will also discuss their limitations and how SpaceLaunch [18] overcomes them in the next section.

3.1. iOS App Launcher

iOS 10 displays have 4x5 grid to place up to twenty apps or folders in a flow layout approach on each page (Figure 1a). When a page's capacity of placing apps is full (i.e., the grid is full), additional apps and folders are placed in the grid on a subsequent page. All apps are placed in the launcher and can exist in only one place. The size of an app or folder is the same, and they are labeled in the same way. They are placed at a specific location and can be moved around by a touch-hold-drag process. Users can select to place a folder or app on a different page even if grid has more capacity left. 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.

Users can place as many apps as they want inside a folder, but folders cannot be placed inside other folders. A folder can have multiple pages where each page contains 9 apps.

(a) IOS 10 Home Screen (b) Folder in IOS 10 (c) Searching in IOS 10

(d) Android home screen (e) Folder in Android (f) Searching in Android

Figure 1. IOS and Android App Launchers

When a folder is opened, the apps on the first page of that folder are presented to the user (Figure 1b). The circles near the bottom of the folder represents the pages inside the folder. Swiping (left or right) or touching on a circle or can take a user to particular page, similar to the main display pages. From the main display, 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.

To search for an app or folder, a user can touch anywhere on the screen and drag down. The search screen has transparent overlay and a search box on the top. Upon typing keywords, all the items (e.g. apps, messages, websites) containing that keyword, show up as a list below the search box (Figure 1c), and the list updates dynamically based on each key press and the contents of the search box. The list has the most used app at the top. If an app is a part of any folder, that folder name is also shown at the right end. Touching on the cancel label to the right of the search box closes the search screen. Selecting any app from the search result list launches the app immediately.

3.2. 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 (Figure 1d). Instead an app drawer with an alphabetical listing of apps is provided. User can place the apps on the launcher page from the app drawer by touching and holding an app icon. Apps are organized in a 4x5 grid on the home screen or a 5x5 grid on subsequent pages. Items can be placed arbitrarily on the grid unlike the flow layout used in iOS.

The quick launch bar at the bottom of the page contains 5 items (and can be dragged upwards to reveal the app drawer). Folders on the main display only show four items; all other items in folders are hidden. When folders are opened, a folder page shows a 4x4 grid of apps (Figure 1e); if there are more than 16 apps in the folder, the behavior of the folders is the same as iOS including the use of circles to indicate and navigate pages.

Android does not provide a search function on the home screen. In order to search for an app, a user has to drag up the app drawer, which has a search field on the top (Figure 1f). The remaining search functionality is functionally equivalent to iOS.

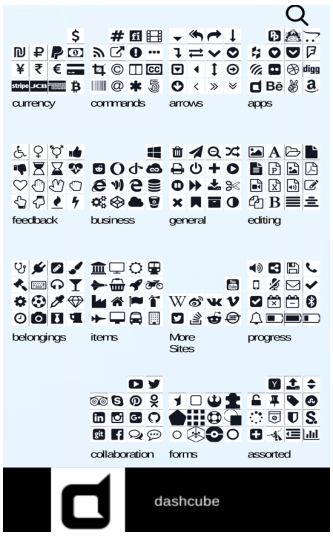
4. SPACELAUNCH

Current app launchers have some limitations. They make it difficult for people to develop their spatial memory by hiding applications inside folders or on multiple pages, which are not visible without slow interactions. Only the apps present on a current page (for the folders - current page inside the folder) are visible. This makes it impossible to perform overall visual search without extra effort (i.e. by swiping, opening folders), which leads to slower interactions. Another drawback is that the same location is occupied by different apps on different pages, which creates conflict within spatial memory.

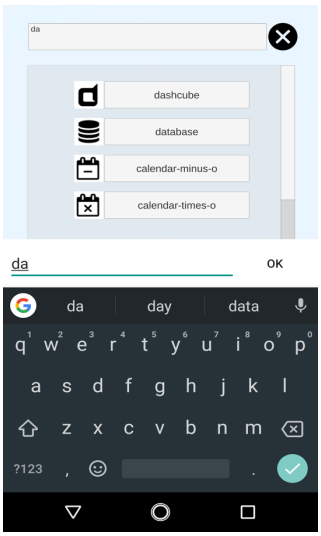
SpaceLaunch has been designed to arrange apps using a flat hierarchy. All the apps are presented at the same level, so that they all are available for visual search without additional steps (i.e., swiping). SpaceLaunch could fit over 200 apps on a single screen. For this study, we used 130 apps, divided then into 14 groups on SpaceLaunch. The placements of groups act as landmarks to guide a user's spatial memory. The groups were labeled with meaningful names suggesting what type of icons are inside the group. However, we also considered the fact that people do not always choose the optimal organization of apps inside folders; it could be randomized. Therefore, we also added some random or irrelevant apps to the groups to make it more realistic. To fit a large number of apps on a single page, the icon size had to be smaller than the usual (as used in Android or iOS). The smaller size could make direct selections difficult and inaccurate; to overcome this problem we provide a tap-to-zoom interaction. The groups can be zoomed in on with a tap; once zoomed in, all the apps in that group are visible but are larger to facilitate selection.

We designed SpaceLaunch to address many of the shortcomings that exist with current app launchers. While SpaceLaunch was informed by the design of several previous spatial interfaces, there were two main questions about our SpaceLaunch implementation that led us to question whether it could compete with current app launcher designs. First, SpaceLaunch uses small icons. If the icons are too small or uncomfortable to use this would greatly slow search times. We need people to search these icons quickly for our design to be effective. Second, SpaceLaunch shows over 100 icons at once. The large number of icons might be overwhelming and difficult to perform visual search to recognize icons. Through our previous study, we demonstrate that SpaceLaunch is faster than the traditional app launcher layouts (either page-based or folder-based) for novices, and that it better supports a transition to expertise [18].

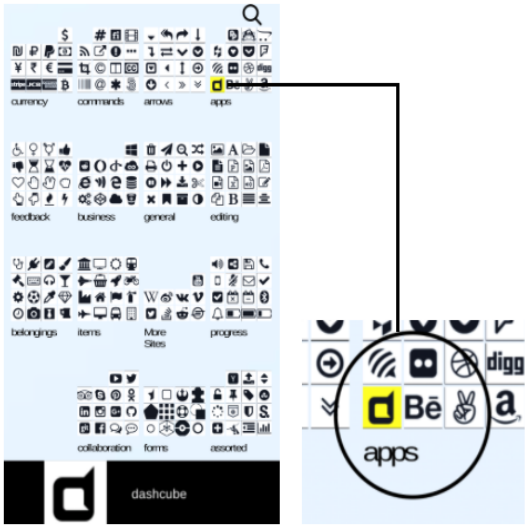
Current app launchers provide efficient and easy to use search functionality; however, it has the shortcoming that it does not allow users to learn the location of an app after successfully searching for it and launching it (i.e., no location learning). SpaceLaunch's search-to-learn functionality addresses this problem by imposing an additional step on the user, allowing them to learn the location of the app for future accesses. When a user types a name of an app, the list shows up with the related suggestions; upon clicking an app the search screen closes, and the user is returned to the main screen, and is able to see the icon they selected highlighted in yellow. This is an effort-inducing interface that requires an additional step but allows the user to learn the exact location of the item. This simple addition to the process of searching for apps should be sufficient to help users to remember that app’s location (see Figure 2).



(a) SpaceLaunch main screen



(b) Searching a target in SpaceLaunch



(c) Searched target highlighted on main screen

Figure 2. SpaceLaunch Search-to-Learn functionality

5. STUDY

In this study, we implemented a modified search functionality which notifies the user with the app location instead of directing the user to the app, which adds a step to the user interface. Our interest was to observe a user's performance with the new search method while comparing it to traditional search.

5.1. Apparatus

We developed an experimental system using Unity 5.6 that provided 2 interfaces of SpaceLaunch (which worked as described above); a search-to-learn interface and a typical interface (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 a Moto Z2 Play (5.5 inches; 1440 x 2560 pixels, 16:9 ratio (~534 ppi density) for both studies.

5.2. Control Condition: Traditional Search

This interface uses a traditional search method which is used in existing app launchers. In our implementation to search for an app, the user clicks the magnifying glass in the top right corner of the screen (Figure 2a). The search screen then appears, with a text box at the top where a user enters a keyword. A list of suggested apps appears in the area below the text box (see Figure 2b). Suggestions are updated with each key press and are displayed alphabetically. Each suggestion item contains an icon and the label of the corresponding app. The user can tap on any item to select an app, and optionally scroll through the list to view all suggestions. Once a user makes a selection, the search screen disappears, and in our system the screen flashes green (a correct selection) or red (an incorrect selection). In an actual app launcher the application would be launched.

Like actual app launchers in iOS and Android, there is no opportunity for the user to learn the location of the app before it is launched.

5.3. Experimental Condition: Search-to-learn

With SpaceLaunch, we present a modified version of the traditional search method that we call search-to-learn. Searching for an app functions exactly as described above. However, upon selecting an app from the search list, the search screen disappears, and the main screen is shown again with all apps visible. The app selected from the search list is now highlighted in yellow (see Figure 2c, where the 'dashcube' icon was selected and highlighted). Finally, the user must select the highlighted app to launch the app, just as they would had they never used search in the first place. This artificially imposes an additional step and effort on the user to help them learn the location of the app. After opening the search screen, if the user wants to go back without making any selection, they can simply click on the X button placed to the right of the text box and they will be taken back to the main screen.

Both interfaces give the freedom to select apps without using search if users know the location of an app already or prefers to use visual search to find it. We closely examine user behavior and performance to use search functionality with both interfaces in the study results described below.

5.4. Icon Sets

We developed 2 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 435 icons. We then grouped these icons into 14 groups that we judged to be most semantically similar. We then divided the larger set into 2 separate smaller sets of 218 and 217 icons, respectively, in 14 groups, each group having its own unique label, containing between 9 and 16 icons each. This meant that our 2 sets were distinct but were balanced in terms of their contents. We also grouped some icons which were not used in either of these sets to create a set for a practice round. It should be noted that we focused a great deal of effort on providing control and consistency for these icon sets; however, there is no perfect grouping of items that would satisfy all use 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.

5.5. STUDY:

The goal of our study was to see if the effort-inducing search-to-learn interface would allow participants to learn app locations and, to assess whether the additional time requirements and effort of learning app locations would reduce the need to search once user starts learning the locations.

Participants

We recruited 32 participants (10 females) from a local university population. Twenty-seven of them were students. Participants averaged 6.06 years of smartphone ownership. When asked how often they used search, 2 participants said they used search frequently to find apps, 23 rarely or never used search, and 5 sometimes used search.

Procedure

Participants were welcomed, explained the study procedure, given an informed consent form and a short demographics questionnaire (focusing on their mobile usage). The study contained two interface conditions (search-to-learn and typical), whose presentation was fully counterbalanced with the 2 icon sets. Before starting the experimental tasks with each interface, we demonstrated each interface with the practice icon set, asking participants to practice until they were comfortable. Participants then started the experimental trials, finding 10 pre-selected (these are referred to as rehearsed icons and are used to assess participant learning) and randomly presented icons (ranging from 1 to 10; these are referred to as non-rehearsed icons and are used as distractors to increase the difficulty and external validity of the experiment) in 10 blocks. Block is an iteration of task performed without any break. In each block, participants were asked to find the icon presented as a target on screen. The total number of targets (now onwards referred as 'trial') per block could range from 11 to 20 depending on the number of non-rehearsed icons that were randomly presented. On an average, participants completed 10 blocks of 16.47 trials per block. We wanted to study participant's learning graph as they encounter the icons repeatedly across blocks, so we chose number of blocks to be 10. 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.

Instructions

Participants were presented a stimulus icon to locate, which appeared at the bottom of the interface (located where the shortcut bar would normally be located on a typical launcher; see Figure 2). Participants were explicitly told that "If you know where an icon is than it will be fastest to select it. However, if you do not know where an icon is located then it will be fastest to use the search functionality rather than looking at it." We had confirmed this instruction to be true through piloting, but we also found in piloting that participants tended to opt for visual search rather than use the search functionality. After this instruction, participants were free to use whichever strategy they liked to find an app.

5.6. 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 number of keystrokes entered; 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.

6. RESULTS

6.1. Data Analysis:

Performance data were analyzed using a 2×10 RM-ANOVA, with interfaces (search-to-learn and typical) and block (1-10) as factors. Post-hoc tests used a Bonneferoni correction to the alpha value (.05) as appropriate. Subjective responses were analyzed using Friedman's test, and Likert-scale responses were recorded as numeric values (0-6, 3=neutral). Free-form text answers are used to illustrate general trends in the findings. Before analysis, outlier trials were removed that were > 3 sd. away from the mean for completion time, this resulted in the removal of 184 trials (2.3%).

6.2. Completion time:

Overall, the avg. completion time taken to complete all blocks using search-to-learn was found to be a bit higher than typical as shown in Figure 3; however, this difference was not significant. There was no main effect of interface on Completion time ($F_{1,23} = 3.974$, ns), means that with search-to-learn interface completion time was higher, but not significantly higher. When we looked at the use of search across all the blocks for both the interfaces, search was accessed more frequently with the typical interface (see Figure 4). In Figure 4, search rate used can be defined as average number of searches used across all blocks.

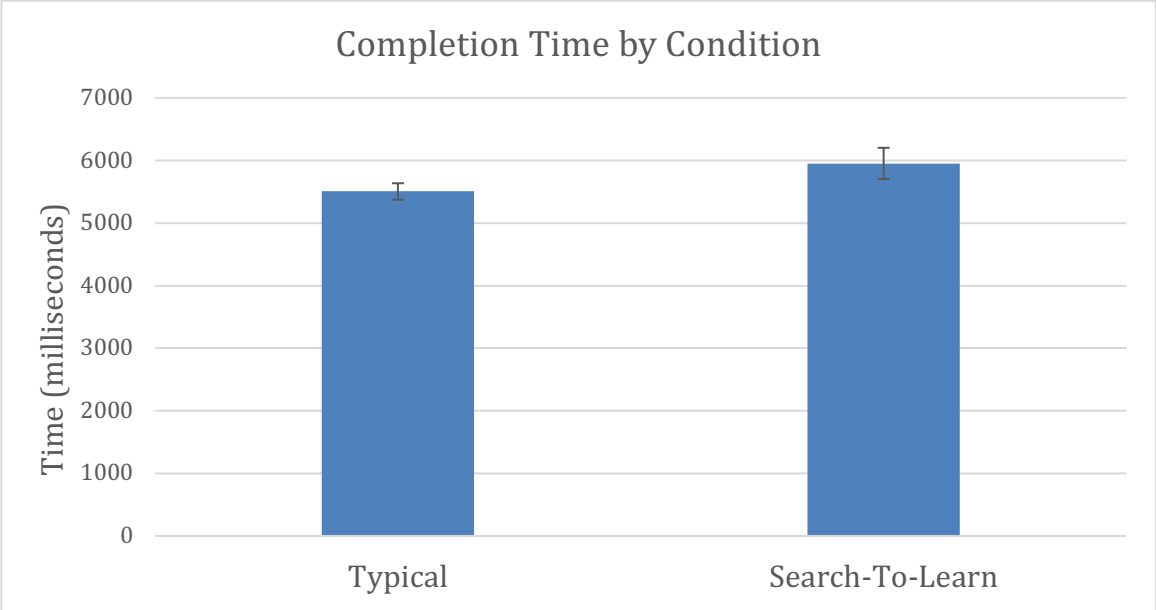


Figure 3. Comparison of avg. completion time by condition

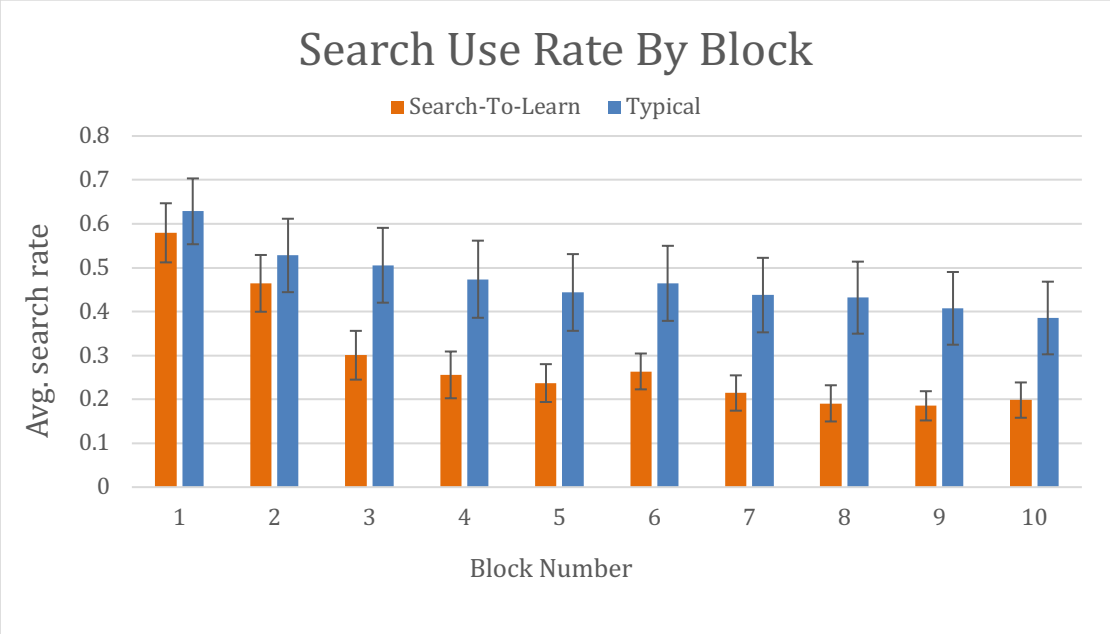
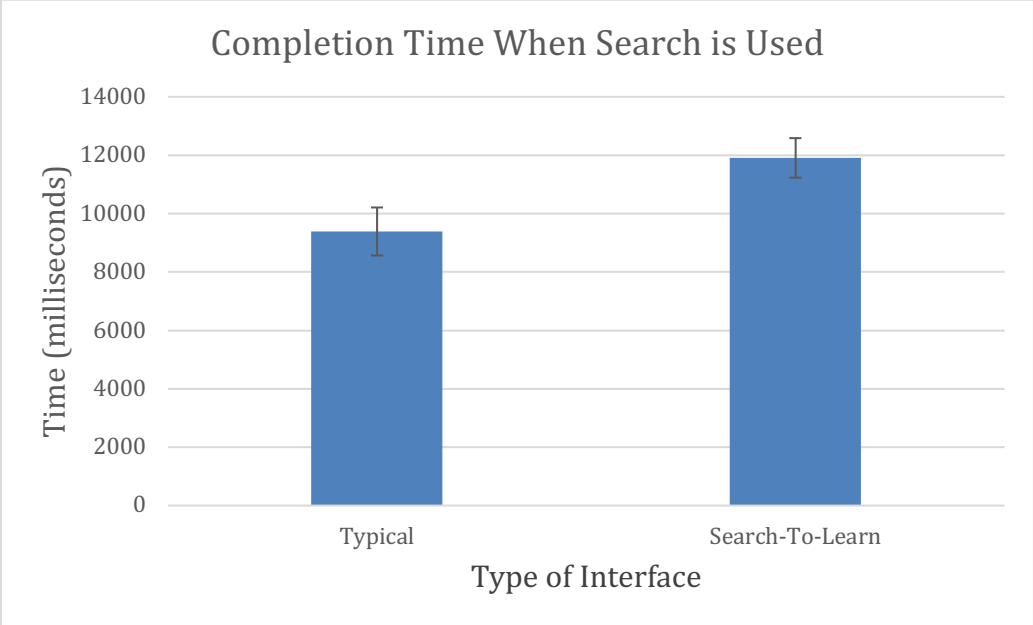


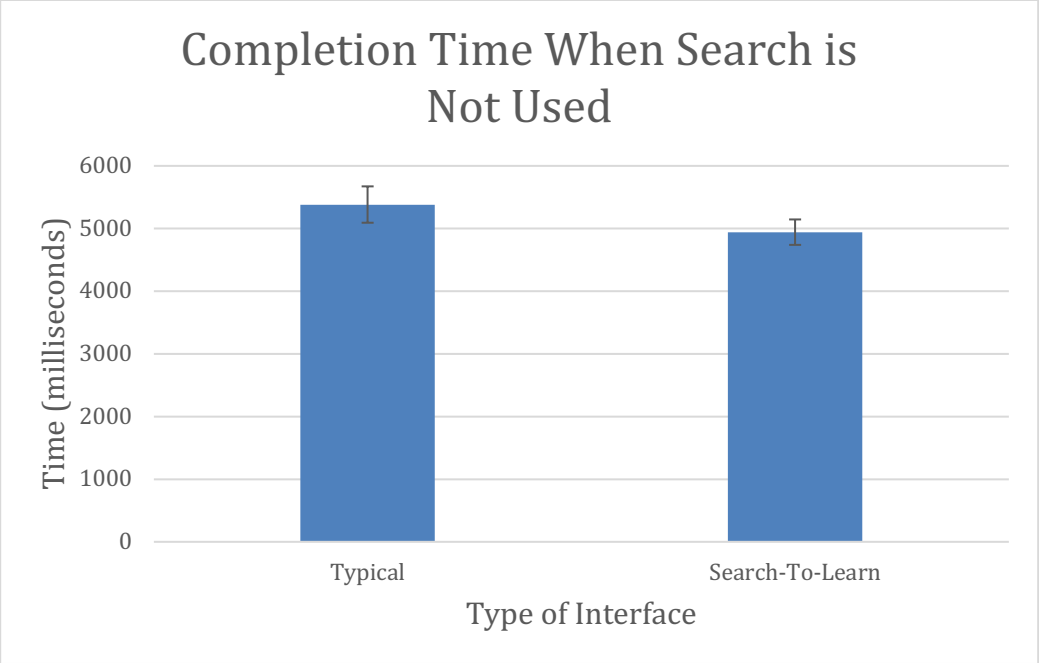
Figure 4. Average search accessed across Blocks

The analysis above included all trials including those when search was not used (i.e., participants accessed the app without using search). When looking at the average

completion time when search was used, search-to-learn was significantly slower than typical ($F_{1,22} = 14.115$, $p<.005$; see Figure 5a). The extra time required with search-to-learn is due to the extra step the user has to perform while selecting a target. When considering trials when participants did not use search, search-to-learn was faster, but the difference was not significant ($F_{1,22} = 3.126$, $p=.09$). While this provides some evidence that search-to-learn helped close the performance gap, it does not provide clear evidence that search-to-learn will always lead to clear and consistent performance improvements (Figure 5b).



(a) Average Completion time when search was accessed.



(b) Average Completion time when search was not accessed

Figure 5. Average Completion time when based on search access.

Since the overall effects included randomly presented (or non-rehearsed targets), it could be that the advantages of the feedback were not enough to make the overall completion time lower. We therefore looked at completion time for both rehearsed (non-random, rehearsed icons) and non-rehearsed (randomly presented distractor icons).

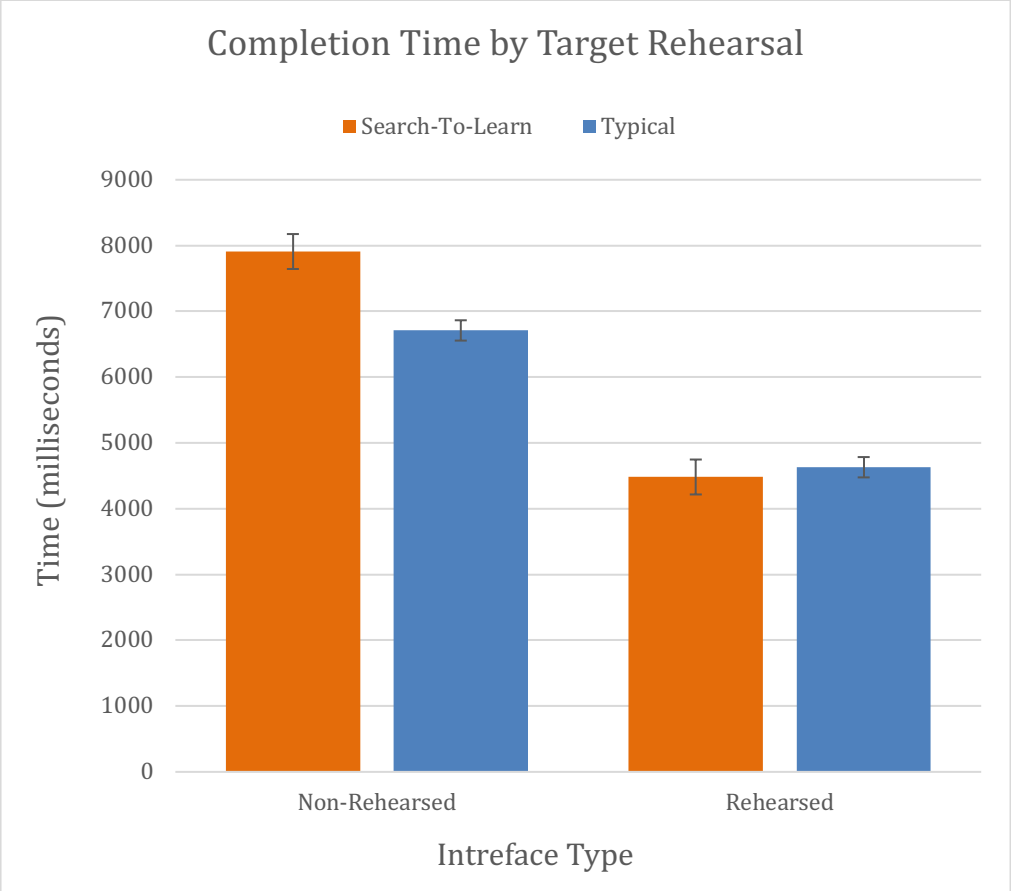


Figure 6. Completion time by Target Rehearsal

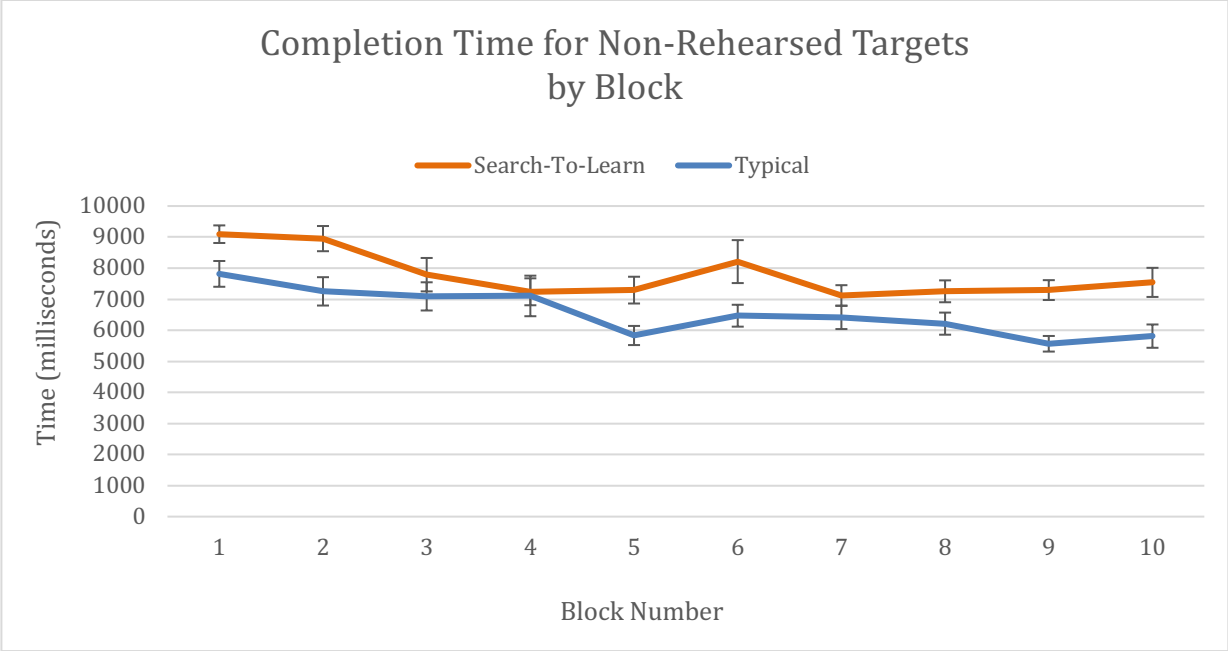
The graph in Figure 6 shows that for the non-rehearsed icons, participants were faster when using the typical interface, and the difference was statistically significant ($F_{1, 23} = 5.265$, $p < 0.05$). There was main effect of rehearsal on completion time ($F_{1, 23} = 140.567$, $p < 0.0001$), meaning that participants were faster for rehearsed targets. There was statistically significant interaction effect of guidance and rehearsal on completion time ($F_{1, 23} = 22.635$, $p < 0.0005$) meaning that participants were faster with search to learn than with the typical interface while searching for rehearsed targets. Interaction effect of guidance means that how guidance provided by search-to-learn affected the completion time. Further Post-hoc comparison showed that there was no significant effect of

guidance for rehearsed icons (($F_{1,23} = 0.591$, ns); whereas guidance had a significant effect for non-rehearsed icons (($F_{1,23} = 13.341$, $p < 0.005$).

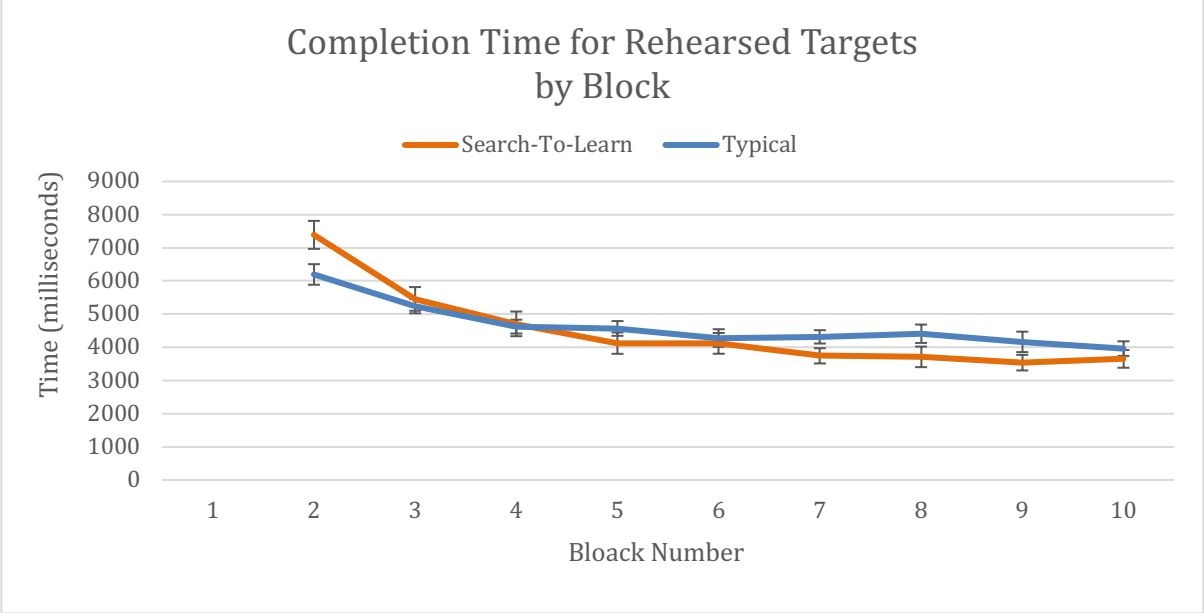
We now examine completion time by block. The block-wise comparison of avg. time taken to find rehearsed targets showed that it takes a little investment of time in the beginning as the targets are new (non-rehearsed); but eventually as the user learns the locations, search-to-learn gets faster than typical interface (Figure 7b). Our block-wise analysis was split by target type (rehearsed or non-rehearsed). However, since participants would encounter each target for the 1st time during block 1, we consider all targets in block 1 as 'non-rehearsed'. Therefore, our block wise analysis of rehearsed targets begins at block 2.

For rehearsed targets, there was no main effect of interface on completion time, meaning search-to-learn was not significantly faster than typical ($F_{1,23} = 0.582$, ns). There was a main effect of block on completion time, meaning that participants got faster as the experiment progressed ($F_{8,23} = 58.144$, $p < 0.0001$). There was an interaction effect of interface and block on completion time ($F_{8,184} = 5.927$, $p < 0.0001$), meaning that search-to-learn improved at a higher rate than typical.

For non-rehearsed targets, there was a main effect of interface on completion time, and search-to-learn was significantly faster than typical ($F_{1, 23} = 14.316$, $p < 0.001$). There was also a main effect of block on completion time, meaning that as the experiment progressed participants became faster regardless of interface type ($F_{9,23} = 8.143$, $p < 0.0001$). There was no interaction effect between block and interface, meaning that the interfaces performed consistently throughout the experiment ($F_{9, 207} = 1.017$, ns).



(a) Comparison of Completion time for Non- Rehearsed targets by block



(b) Comparison of Completion time for Rehearsed targets by block

Figure 7. Comparison of Completion time based on target rehearsal by block

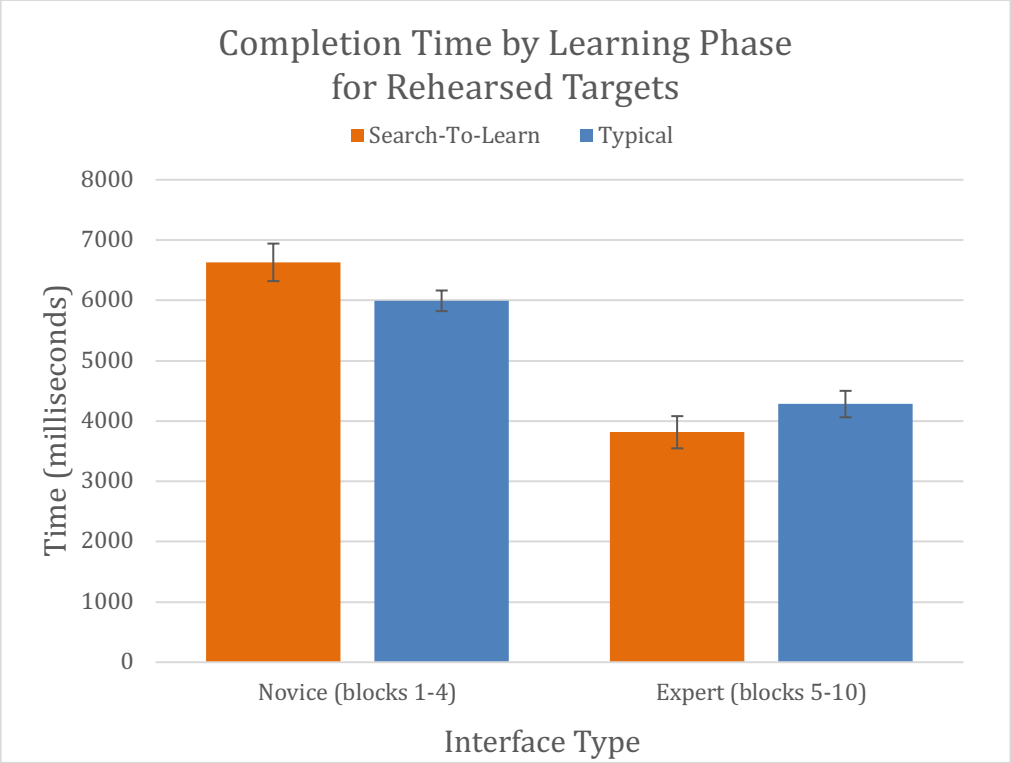


Figure 8. Comparison of Completion time by learning phase

We used the by block analysis to examine the moment when users tended to transition from novice to expert modes of access, that is the point in time when their performance becomes better with search-to-learn than with typical. Visually analyzing the by-block performance for rehearsed targets, we can see that it takes roughly 4 rehearsals before participants consistently perform better with search-to-learn (see Figure 7b). We therefore analyzed overall performance between these two phases (novice – blocks 1-4 – and expert – blocks 5-10). The difference between the novice and expert phases was statistically significant ($F_{1,23} = 193.970$, $p < 0.00001$). Overall, there was no main effect of the interface on completion time ($F_{1,23} = 0.151$, ns). However, there was an interaction effect between interface and expertise on completion time ($F_{1,23} = 11.380$, $p < 0.005$), meaning that search-to-learn did perform better for expert levels than for novice levels.

A pairwise comparison shows that there was no significant difference for novices between search-to-learn and typical ($F_{1,23} = 3.428$, ns.). However, for experts, search-to-learn did perform significantly better than typical ($F_{1, 23} = 6.937$, $p < .05$).

6.3. Subjective Data:

When asked "which interface was the best for finding apps", participants overwhelmingly chose search-to-learn (21 chose search-to-learn, 3 chose typical). As shown in Figure 9, 22 participants said that they were fastest using 'With Guidance', 2 thought 'No Guidance' was faster ($\chi^2 = 15.042$, $p < 0.0005$). Participants felt that search-to-learn was the "easiest to remember target locations" ($\chi^2 = 18.375$, $p < 0.0001$).

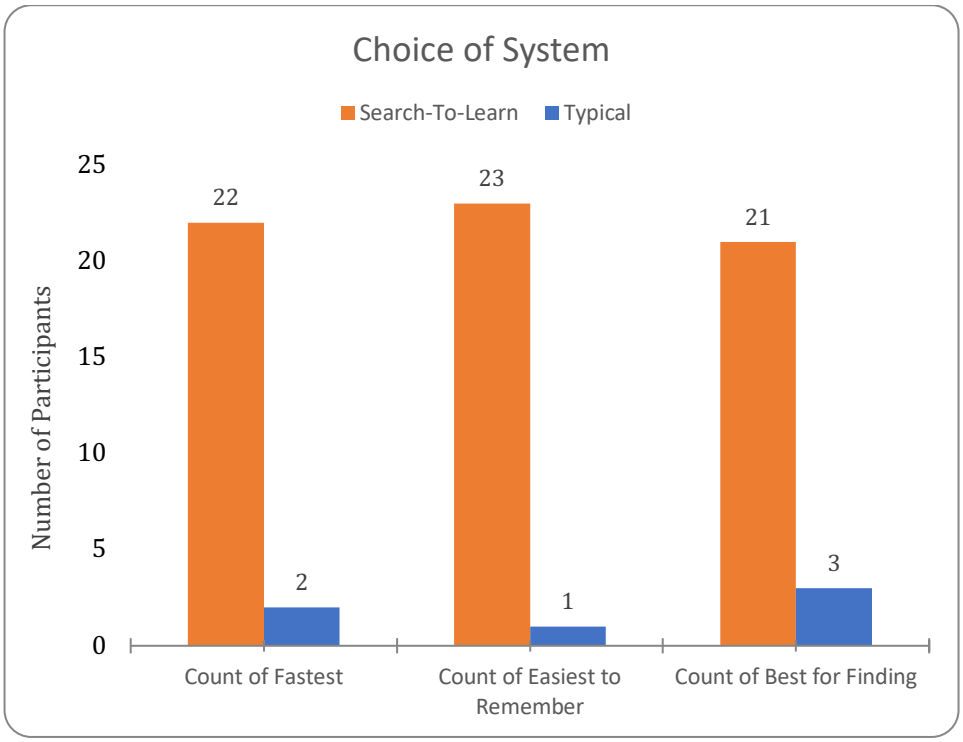


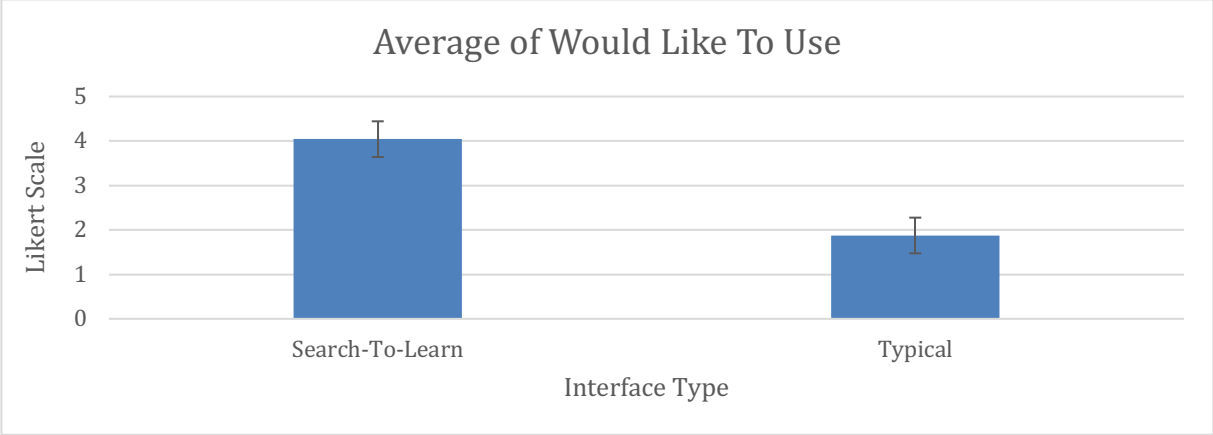
Figure 9. Participants' Choice of System

Most participants also felt that search-to-learn to be best for finding locations ($\chi^2 = 12.042$, $p < 0.001$). One of the participants said, *"It was nice to have the apps pointed*

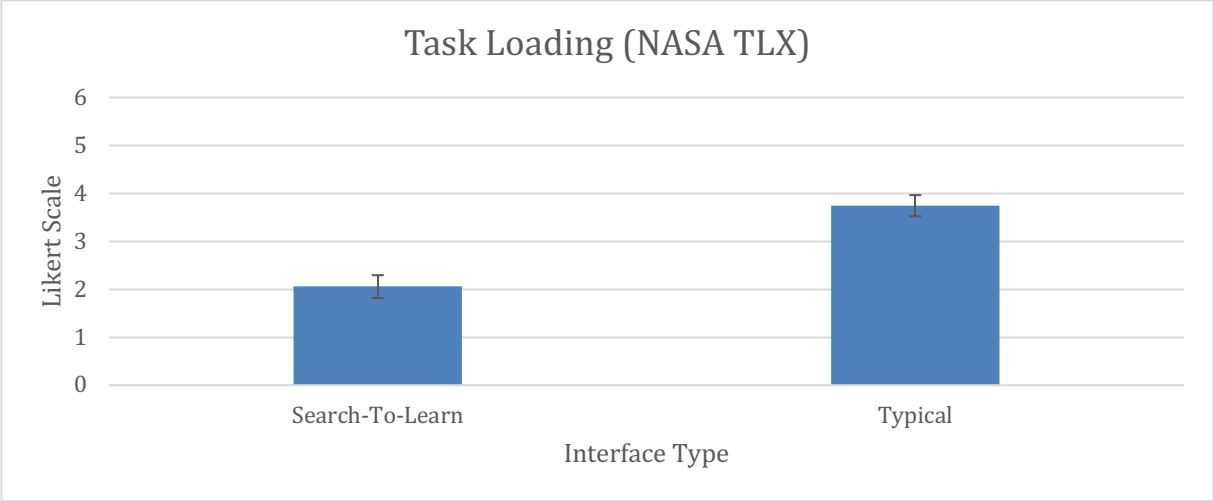
out directly with guidance, so I felt like I had a better chance knowing where they were the next time I had to find them."

While the ratings overall were strongly in favor of search-to-learn, comments provided by participants gave some further contextualization: "*[with search-to-learn] I could actually learn where apps were instead of hoping that I'd be asked to select an app that I happened to see the location... But I probably wouldn't use it in real life--I would just organize my apps better...*" This raises an interesting question. While some participants were warm on SpaceLaunch others were not sure they would want such a different app launcher in practice. We discuss this further in the discussion below.

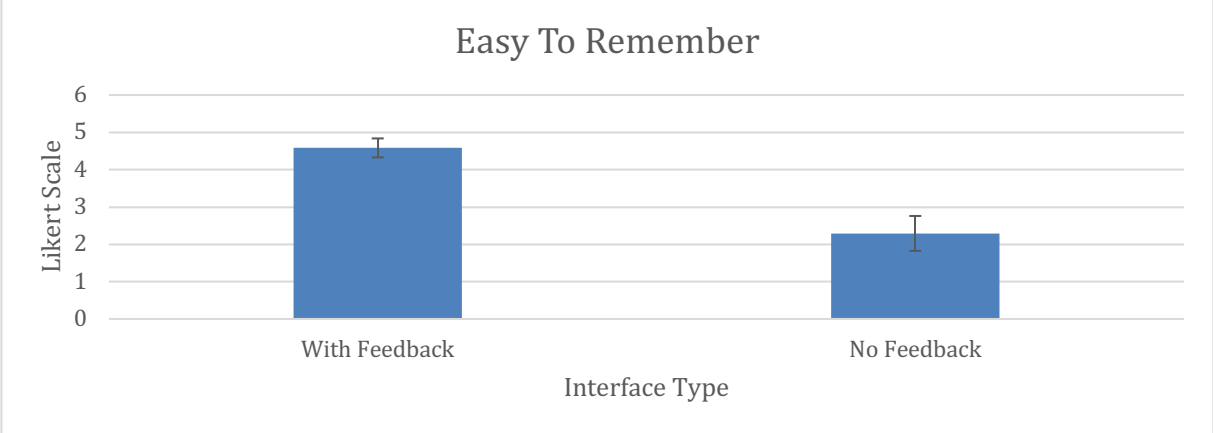
We also collected participant ratings for ease of remembering app locations, user preference and difficulty (via NASA TLX) based on Likert scale ratings (rating between 0 and 6; Figure 10).



(a) Agreement with Would Like to use (higher is better)



(b) Difficulty Level NASA TLX (lower is better)



(c) Easy to remember (higher is better)

Figure 10. User Preference, Task Loading and Ease of remembrance

Participants rated search-to-learn for "I would like to use for finding apps" ($H = 10.714$, $p < 0.005$). The majority of participants rated search-to-learn easier than typical to remember target locations ($H = 15.696$, $p < 0.0005$). Participants rated typical higher for task loading (difficulty) than search-to-learn ($H = 14.222$, $p < 0.0005$).

7. DISCUSSION

Overall, our results are in favor of search-to-learn. Considering a small time investment required for novice to learn app locations, search-to-learn was faster during the expert phase. Participants also seemed to find search-to-learn easier and a better interface for finding apps than the typical search interface when paired with a spatial app finder.

Why did search-to-learn work?

The results showed that the extra steps initially required by search-to-learn was a worthwhile investment because the spatial memory it allowed participants to develop more expert means of access, based on memory to locate and launch apps. While typical search was initially faster, search-to-learn worked better after just 4 accesses of an app. So, participants spent more of the experiment performing better with search-to-learn than they did with typical search.

Why did participants think it required less effort?

While using typical, participants had fewer opportunities to learn where app icons were located, which means there was little-to-no spatial learning. So, even after the icons appear multiple times, participants could not learn where the app was placed. As the participants were able to remember the app locations, they did not have to use search every time, which means their physical and cognitive efforts were reduced. This is one of the reasons why search-to-learn was preferred by the majority of participants.

Would using search-to-learn be worth it?

Our results show that participants were faster in selecting apps with search-to-learn once they became experts. While there was an initial cost to acquiring this expertise, it should

be noted that once acquired, users would spend a longer time as experts. Thus, we conclude that providing some type of guidance, while initially taking more time, would be beneficial in the long term for users for easier and faster mobile interactions.

8. FUTURE WORK

Our study shows that search-to-learn is a more appropriate search interface for the spatial app launcher, SpaceLaunch. Our study has certain limitations that provide directions for future work.

All the icons used in SpaceLaunch are monochromatic (black and white). The reason for not using realistic icons is to prevent color from being a differentiating element for a user. We needed our participant to remember the icons by their positions, not any other factor. In real world, app icons have a different color, size or shape. It would be interesting to study how these factors affect spatial learning with SpaceLaunch.

SpaceLaunch had 130 icons displayed on screen for our study. This number would change based on a user's requirements (e.g., space) and preferences. As SpaceLaunch fits all the icons on a single screen, the icon size will change based on the number of icons. Further work can be done to see how SpaceLaunch works with a different number of icons with different sizes. Does search-to-learn still work better than a typical interface with different icon layouts?

We conducted the study in a controlled environment where participants were asked to access apps repeatedly. Further, exactly how expertise might decay over time (i.e., how long will users remember app locations?) and what expertise might form in real world use (i.e., how many rehearsals will it take to remember locations under real world conditions?) are open questions. To see how SpaceLaunch and search-to-learn might perform in real world use, we would need to develop a real space launch app launcher and collect real world usage data.

9. CONCLUSION

In this thesis, we introduced a new search paradigm, search-to-learn, that assists users to learn app locations in the spatial app launcher, SpaceLaunch. SpaceLaunch is a novel app launcher that supports the rapid finding of apps and the development of spatial memory. While typical search approaches could be used effectively with SpaceLaunch, we created search-to-learn to demonstrate how search can be better aligned with the goals of spatial interfaces. Search-to-learn works by guiding users to app locations, helping them to build spatial memory.

We compared search-to-learn with the conventional search method, referred to as ‘typical', and demonstrated that search-to-learn helps participants reach expertise more rapidly. Once expertise is reached, search-to-learn is faster than typical search. When using SpaceLaunch, participants strongly preferred search-to-learn and thought it was faster and easier to use than typical search.

While the search-to-learn approach does initially require additional time investment during the novice phase, by helping users rapidly learn app locations, it reduces the need to access search, allowing apps to be accessed rapidly accessed via the app’s location on the screen. Our findings provide an important interaction paradigm for spatial interfaces and provides further evidence that the use of spatial memory as a guiding principle in the design of interactive systems is a viable and promising approach.

REFERENCES

1. Andy Cockburn, Bruce McKenzie. Evaluating the effectiveness of spatial memory in 2D and 3D physical and virtual environment. *Proceeding CHI '02 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems Pages 203-210*
2. Andy Cockburn, Carl Gutwin, Joey Scarr, and Sylvain Malacria. 2014. Supporting Novice to Expert Transitions in User Interfaces. *ACM Comput. Surv. 47, 2, Article 31 (November 2014), 36 pages. DOI:* https://doi.org/10.1145/2659796
3. Andy Cockburn, Per Ola Kristensson, Jason Alexander, Shumin Zhai. Hard Lessons: Effort-Inducing Interfaces Benefit Spatial Learning. *Proceeding CHI '07 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Pages 1571-1580*
4. 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
5. Benjamin B. Bederson , Angela Boltman. Does Animation Help Users Build Mental Maps of Spatial Information. *Proceeding INFOVIS '99 Proceedings of the 1999 IEEE Symposium on Information Visualization Page 28*
6. Czerwinski, M., van Dantzich, M., Robertson, G. and Hoffman, H. The Contribution of Thumbnail Image, Mouse- Over Text and Spatial Location Memory to Web Page Retrieval in 3D. *in Proc. INTERACT'99, (1999), 163-170.*
7. Donskoy, M., & Kaptelinin, V. (1997). Window Navigation With and Without Animation: A Comparison of Scroll Bars, Zoom, and Fisheye View. *In Proceedings*

of Extended Abstracts of Human Factors in Computing Systems (CHI 97) ACM Press, pp. 279-280.

8. 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.*
9. 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*
10. 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
11. Jiang, Y., Song, J.H. and Rigas, A. High-capacity spatial contextual memory. *Psychonomic Bulletin & Review 12, 3 (2005), 524-529.*
12. Joey Scarr, Andy Cockburn, Carl Gutwin, Sylvain Malacria. Testing the Robustness and Performance of Spatially Consistent Interfaces. *Session: Spatial Interface*, *CHI 2013: Changing Perspectives, Paris, France*
13. Kevin Lynch. 1960. The image of the city. *MIT Press.*
14. Md. Sami Uddin, Carl Gutwin. The Effects of Artificial Landmarks on Learning and Performance in Spatial-Memory Interfaces. *Proceeding CHI '17 Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems Pages 3843-3855*
15. Monica Tavanti, Mats Lind. 2D vs 3D, Implications on Spatial Memory. *DOI:*

10.1109/INFVIS.2001.963291 Source: IEEE Xplore Conference: Information Visualization, 2001. INFOVIS 2001.

16. Antti Oulasvirta., Mikael Wahlstrom, and K. Anders Ericsson. What does it mean to be good at using a mobile device? an investigation of three levels of experience and skill. *International journal of human-computer studies 69, 3 (2011), 155–169.*
17. Joey Scarr, Andy Cockburn, Carl Gutwin and Andrea Bunt. Improving command selection with CommandMaps. *Proc. CHI'12, ACM, Austin, Texas, 2012, 257-266.*
18. Scott Bateman, Carl Gutwin, Manasi Shah, Nathaniel Brewer. Supporting Visual Search and Spatial Memory in a Mobile Application Launcher. *Technical Report, TN18-240. Faculty of Computer Science, University of New Brunswick. October 10, 2018. 13 pages.*
19. Svenja Leifert The Influence of Grids on Spatial and Content Memory. *Proceeding CHI EA '11 CHI '11 Extended Abstracts on Human Factors in Computing Systems Pages 941-946.*

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 (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering), 294–313

CURRICULUM VITAE

Candidate's full name	:	Manasi Bhagwati Kumar Shah
Universities attended (with dates and degrees obtained)	:	MCS by Report (in progress) University of New Brunswick (2017-Present) B.E. Computer Science Gujarat Technological University (2010-2014)
Conference Presentations	:	Science Atlantic Conference 2017 Title - SpaceLaunch: A faster app launcher that supports spatial memory October 13-15, 2017, University of New Brunswick, Fredericton (Received the 'Science Communication Award')