BinarySwipes: Fast List Search on Small Touchscreens

Johannes Hartmann
Maximilian Schirmer
WeLoveApps GmbH
Erfurt, Germany

Florian Echtler
Bauhaus-Universität Weimar
Faculty of Media
Weimar, Germany

ABSTRACT

Smartwatches and other wearables generally have small screens, thereby complicating touch-based interaction. Selection from a long list, e.g. to locate a contact or a music track, is particularly cumbersome due to the limited interaction space. We present BinarySwipes, an interaction technique based on binary search which is designed to speed up list search tasks on space-constrained screens. We evaluate a prototypical implementation of BinarySwipes on a smartwatch. Results from our evaluation with 21 participants show improved performance over a plain linear search on lists with 100, 200 and 500 entries, but also increased mental load on the users.

CCS CONCEPTS

• **Human-centered computing** → **Touch screens**; *Gestural input*; Ubiquitous and mobile computing systems and tools; *Empirical studies in ubiquitous and mobile computing*.

KEYWORDS

smartwatch; touch screen; wearable; search; list search; touch interface; small screen

ACM Reference Format:
Johannes Hartmann, Maximilian Schirmer, and Florian Echtler. 2019. BinarySwipes: Fast List Search on Small Touchscreens. In *Mensch und Computer 2019 (MuC '19), September 8–11, 2019, Hamburg, Germany*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3340764.3340774

Figure 1: Binary search interface running on an Apple Watch v2.

1 INTRODUCTION

Wearable devices such as smartwatches offer severely limited screen space, even though smartphone screens have constantly been increasing in size for the last few years. Consequently, most touch-based interaction techniques such as pinch-zoom assume that a sizeable amount of screen space is available, and are therefore difficult to transfer to the limited screen sizes of such small devices.

One example for such an interaction technique is scrolling through and searching in a list of items, such as contacts or song titles. On a smartphone, this is usually solved through a combination of inertial scrolling, a draggable position indicator on the side, and/or a search field. None of these transfer well to a small screen: the relevant item is more easily missed during inertial scrolling, a position indicator is either hard to hit or occupies too much screen space, and efficient text entry on small screens is an unsolved problem on its own.

To address this challenge, we introduce *BinarySwipes*, an interaction concept that uses the fundamental paradigm of binary search to improve list search on small touchscreens.

MuC '19, September 8–11, 2019, Hamburg, Germany

ACM ISBN 978-1-4503-7198-8/19/09…$15.00
https://doi.org/10.1145/3340764.3340774

BinarySwipes does not rely on any physical controls and only requires a standard touchscreen. We implemented a prototype of this concept on a smartwatch, and evaluated the prototype with 21 users in a controlled study. A noteworthy result from our evaluation is that BinarySwipes improves raw search performance, but also increased mental load in comparison to a plain linear list interface.

2 RELATED WORK

Several existing research projects have already focused on how to enable efficient interaction with a small touchscreen in general. For example, Lafreniere et al. [8] looked into how more complex tap gestures could be used to improve command selection on smartwatches, with encouraging performance gains reported in their study.. Singh et al. [15] recently investigated how increased mobility and encumbrance of users affect performance on such devices. Their findings suggest that short-contact gestures such as flicks are best suited for this interaction context.

Regarding list interaction, one of the most commonly used lists on smartphones and smartwatches is the contact list. Bentley and Chen [2] analyzed the mobile phone books of 200 users, and found an average size of the contact list of 308 entries (with an astonishing maximum of 3038 contacts). Obviously, finding an entry in lists of this size would be a very tedious and time-consuming task on a space-limited touchscreen interface when using a plain list.

Absolute Positioning

In the context of list selection, Quinn and Cockburn [12] present *Zoofing!*, an approach to scrolling that uses flicking and pressure as possible ways to scroll and "zoom" in lists. The initial list of items (the paper also lists contact and song names as examples) is presented completely zoomed out, i.e. every item in the list is visible. So-called landmarks at the side of the list denote the start of sections, e.g. the first letter of a term in an alphabetic list. Putting pressure on the screen activates the zooming part of the interface, with zoom level proportional to increasing pressure. If the user eases the pressure, the zooming stops and eventually returns to a fully zoomed out state after three seconds without interaction. After the zooming phase, flicking the screen scrolls the interface with simulated friction. Once the user reaches the intended list entry, tapping can be used for item selection (but is active at any given zoom level). The user tests the researchers performed showed a significant improvement of item acquisition times as opposed to traditional approaches.

Perrault et al. [10] use a custom smartwatch wristband with embedded sensors to detect interaction on the band. Position on the wristband is used to indicate absolute position in a list, which their study shows to outperform a plain scrolling interface. A fundamentally similar approach is followed by Ahn et al. [1], who embed touch-sensitive edges into the rim of a square smartwatch that can then be used for scrolling, similar to the edges of many laptop touchpads. Corsten et al. [4] use a force-sensitive touchscreen to augment value selection (e.g. in a date picker), while Xiao et al. [17] convert the entire front face of the watch into a physical button that can be used for navigation. However, a common drawback of all these approaches is that they require custom sensor hardware in the form of a sensor-equipped wristband, touch-sensitive casing, or force sensors, and can therefore not directly be generalized to other small-screen devices which only offer a plain touchscreen.

List Segmentation

Rao et al. [13] present a list interaction concept which iteratively partitions a list into multiple sublists based on user input, an approach which can be considered a generalization of binary search. Similarly, Shani et al. [14] introduce a technique called *Character Pinning*. In an interface that displays a search term, the user can either accept the current word or indicate that the searched-for word is alphabetically above or below the current word. In addition, pinning a character locks the first letter of the word and search continues with a subset of words which start with the corresponding letter. This equals the first step of the hybrid search option of our approach discussed below. Subsequently pinning more characters will further expand the prefix and thereby lower the amount of possible selections left. The author's user study found significant advantages when used on a button-based interface. We follow a similar approach with respect to the underlying fundamental concept of binary search, but extend the interface using range indicators (see below) and investigate its usability on a space-constrained touchscreen device.

In a similar way, *CircularSelection: Optimizing List Selection for Smartwatches* by Plaumann et al. [11] uses a ring-shaped array of starting characters around the periphery of a circular smartwatch in order to pre-select a sublist, which is then accessed in a regular manner. The authors found significantly faster search times with respect to a plain list interface on lists with 40 and 240 items.

Based on the earlier "effective view navigation" by Furnas et al. [6], Chittaro et al. [3] present an approach to filtering long lists by not showing an entire list, but splitting the long list into multiple smaller ones and then displaying the first and last entries of the respective list. This essentially changes the search algorithm from a linear search into a tree search, as the respective sub-lists can again be split into smaller

arrays. An interface using the effective view navigation always offers to either select one of the displayed words or to dig deeper into the arrays that lie between the words that are currently displayed. The authors use this approach on a smartphone-sized device and tests its effectiveness with two variants, a split into two arrays and a split into six arrays. They compare these two interfaces with a traditional keyboard and search field approach. While the evaluation shows that this technique is not yet capable of competing with regular search via keyboard, they still are convinced that this technique "could still be useful when no full keyboard [...] can be provided". As one of the use cases they also name scenarios in which wearable computers are relevant, which applies to our present work.

Summary

From reviewing related work, we conclude that there are few techniques for interaction with long lists that are applicable to a plain, small-scale touchscreen which a generic smartwatch provides. Navigation concepts which are based on binary search show promise, but have not yet been modified to fit this context. We combine this approach with a suitable input method focused on coarse swipes, resulting in our *BinarySwipes* prototype detailed below.

3 BINARYSWIPES PROTOTYPE

Our *BinarySwipes* prototype (shown in Figure 2a) implements an interaction concept based on binary search, i.e. the user is presented with a currently selected "pivot" element (the center of the currently selected range), and is given the option of a) navigating to the sub-range lexically prior to the pivot element, b) navigating to the subrange lexically subsequent to the pivot element, or c) confirming the currently selected element and thereby ending the search. At the start of the search, the entire list contents are the selected range, and the middle element of the whole array is the pivot element. To select the lexically prior/subsequent subrange, the user needs to swipe up/down, while confirmation is performed through a dedicated touch button.

The prototype described here is the second iteration of our interface. It is based on the results of a preliminary qualitative user test with 6 participants, which led to the introduction of the range indicators and the hybrid search mode described below.

Undo/Reset Buttons

If the user accidentally swipes away from their target word, the wrong sub-range gets picked as the new focus. This array does not contain the word the user initially was looking for, at which point the user-driven algorithm breaks. This requires the interface to offer an undo button. Furthermore, the interface has also been given a full reset option. If the user does not immediately realise that they made a mistake, resetting the interface can be less daunting than undoing an unknown number of steps.

Range Arrays and Prediction Indicators

The upper and lower array indicators are a visual clue inspired by earlier work from Chittaro and De Marco [3]. Without any form of indicators, especially with names, the interface forced the user to think about the alphabet sequence at every step. To mitigate the implied mental load, the range indicators were added.

Above and below the current word or letter, the interface now displays the upper and lower range's boundary elements. Thereby, one look could possibly suffice to tell in which direction the searched letter or word is to be found. In addition, once the user swipes up or down slightly (not triggering the full swipe yet), the interface will display which letter/range the swipe will lead to.

Linear and Hybrid Search Variants

For comparison, a "regular" linear list search with swipe-based inertial scrolling and a side position indicator was also implemented, using the default list widget on the watch. Confirmation is performed by simply tapping the correct list entry. The scale was chosen so that three different names could be displayed simultaneously, based on e.g. the default calendar view on the Apple Watch which also shows 3 items per screen (see Figure 2c).

In addition, we implemented a hybrid search step in which the user is first supposed to select the starting letter of the search term using binary search (see Figure 2b). Afterwards, they are presented with a linear sub-list of the initial items which exclusively contain words that start with the selected letter. Consequently, the user can scroll through smaller lists throughout the entire process, and the mental load might possibly be lower.

Additional Implementation Aspects

Although our prototype was implemented on an Apple Watch which also offers a digital crown for scrolling interactions, we decided against integrating the crown into our interface. The rationale for this choice is that the crown, or similar features such as rotateable bezels, only are available on a limited subset of wearable devices, whereas a touchscreen is nearly ubiquitous. Therefore, we focused on interaction mechanisms that could comfortably be implemented on any small touchscreen device.

In addition to the central watch implementation, we built an iPhone application that is used in later tests to reset the application on the watch, for switching between the different search modes, and for logging task completion times and error rates.

Figure 2: Screenshots from the watch simulator showing the three possible interface modes: (a) shows the binary search, (b) the hybrid mode and (c) the linear search.

4 EVALUATION

To test the effectiveness of the interface presented above, we conducted a controlled within-subject user study that tested the viability of the binary search interface in comparison to a common linear list interface. In total, we evaluated the BinarySwipes interfaces with 21 users. Their average age was 26.1 years (σ = 3.6 years). Eight of the users had a computer science background, while the rest came from various backgrounds ranging from physical health care to political sciences. Participants were provided with refreshments, but did not receive monetary compensation. After answering a short demographic questionnaire and consenting to the aggregated and anonymized use of their data, we introduced them to the smartwatch interface.

We tested three interface conditions:
Condition 1: **Linear**. As described above, this baseline condition uses the standard list widget available directly from the watch UI framework, and supports dragging/flicking interaction to scroll the list.
Condition 2: **Binary**. This condition uses swipe-controlled binary search through the full list until the correct entry has been found and selected via the confirmation button.
Condition 3: **Hybrid**. This condition combines a binary search step to select the starting letter of the search word, followed by a linear search through the respective sublist of words starting with that letter.

The three conditions were presented in counter-balanced order. This is especially relevant because a majority of the test users (15) said they have no prior experience with smartwatches at all. Therefore, learning effects might not appear with respect to the interfaces themselves, but rather with respect to using and wearing a smartwatch in general. Note that we did not choose a full Latin square randomization of the three individual conditions in order to avoid more than one interface context switch (linear vs. binary/hybrid) per participant. Consequently, the condition orders in our experiment were LBH, LHB, BHL, and HBL.

Each interface condition was tested with three different list lengths: 100 (small), 200 (medium) and 500 (large) fictional names. These list lengths were presented in randomized order to prevent learning effects. For each list length, the user was tasked with finding three randomly selected names, displayed on the controlling smartphone in front of the participant. For the linear list with 500 entries, we unfortunately experienced transient performance issues and crashes with the Apple Watch 2 which resulted in a noticeably smaller data set for this condition. Since the binary and hybrid interfaces do not have to keep all 500 names active in a single list widget, the 500-item condition did not cause any problems for these conditions.

After finishing all three runs through each interface condition, the users were presented with a NASA-TLX questionnaire to provide qualitative measures on the interface. Finally, the users were invited to voice any ideas, thoughts or concerns with the interface that they had.

During each test, we logged the total time required to find the name in question as well as the number of undo and reset actions in the binary interface. The number of undo steps was only tracked in for the binary search conditions because the linear search does not explicitly support this operation.

Results

For all statistical tests, we use the standard confidence interval of 95% for determining significance.

Each test run resulted in a total of 63 samples per interface and length of the array, making for a total of 9 x 63 = 567

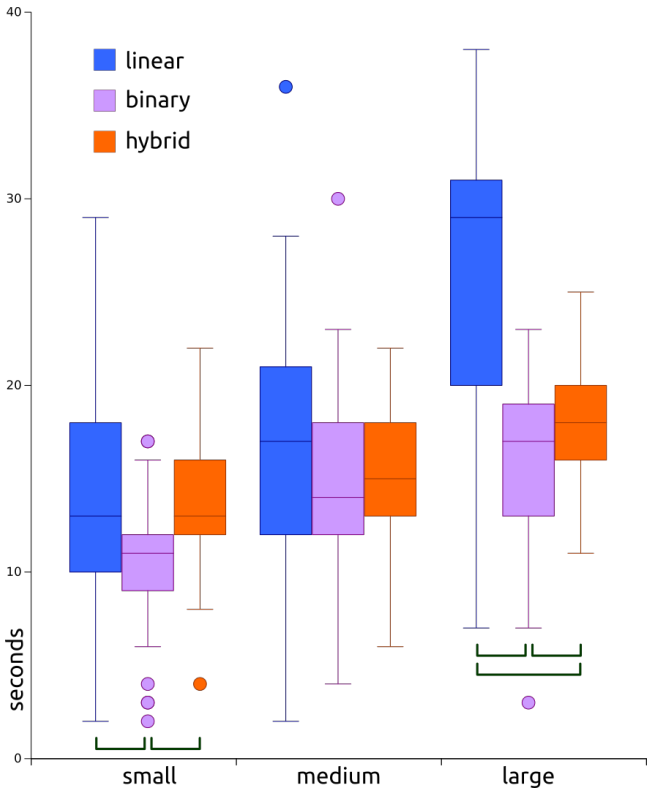

Figure 3: Search times for each condition and list size. Brackets indicate significant differences between conditions.

data points. Due to transient performance issues in the linear-large condition (possibly due to memory exhaustion caused by the 500-item list), 34 invalid samples had to be removed. Consequently, results involving this combination have to be treated cautiously, although we still included these in the analysis. Each linear search showed a normal distribution according to the Shaprio-Wilk test (for p = 0.05; w = 0.956 with small, w = 0.985 with medium, and w = 0.939 with large list). Similarly, the binary search showed normal distribution (for p = 0.05; w = 0.948 for small, w = 0.950 for medium, and w = 0.963 for large list), just like the hybrid search (for p = 0.05, w = 0.950 for small, w = 0.964 for medium, and w = 0.972 for large list). Consequently, as all completion time data are distributed normally (as also confirmed by visual analysis of density and QQ plots), this enables the use of ANOVA with post-hoc Tukey HSD test for normally distributed samples for any comparisons of the data groups.

On average, users took 14.1 seconds ($\sigma = 5.78s$) for finding a name in the regular search with a small array, while the time increased to 16.6 seconds ($\sigma = 6.61s$) for the medium array, and to 25.5 seconds ($\sigma = 7.90s$) with the large array. The binary search was faster in absolute terms, taking 10.6 seconds on average ($\sigma = 3.09s$) on a small array, 14.8 seconds ($\sigma = 4.51s$) on a medium array and 15.6 seconds ($\sigma = 4.32s$) for the large array. The hybrid search took a little longer on average with a search taking 14.2 seconds ($\sigma = 3.74s$) on the small list, 15.6 seconds ($\sigma = 3.52s$) on the medium list and 18.0 seconds ($\sigma = 3.36s$) on the longest array. Figure 3 shows an overview of all time measurements.

ANOVA shows that the results differ significantly for the small ($p < 0.001$) and large ($p < 0.001$) conditions, but not for the medium condition ($p = 0.141$). Table 1 shows the p-values for all valid comparisons, i.e. between same array sizes (as provided by the post-hoc Tukey HSD test).

In particular, the binary search performed better than the linear search by 3.52 seconds for the 100-item condition. This difference is significant based on the ANOVA result ($p < 0.001$). For 500 items, the difference of 9.90 seconds is also again significant at $p < 0.001$. Interestingly, all three tests show an absolute time advantage towards the simpler version of the interface (without hybrid search). The comparison for the 100-item condition shows a significant difference ($p < 0.001$), as does the comparison for the 500-item condition ($p = 0.013$), suggesting the regular binary approach as a better choice for these array sizes.

Cond. 1	Cond. 2	p-value	Implication
Linear S	Binary S	< 0.001	Binary S is faster
Linear S	Hybrid S	0.998	No significant diff.
Hybrid S	Binary S	< 0.001	Binary S is faster
Linear L	Binary L	< 0.001	Binary L is faster*
Linear L	Hybrid L	< 0.001	Hybrid L is faster*
Hybrid L	Binary L	0.013	Binary L is faster*

Table 1: All Tukey post-hoc comparisons for significantly different conditions (as indicated by ANOVA). Results marked with ∗ are based on reduced sample size due to transient performance issues.

We see the following absolute differences between linear and hybrid search: -0.05 seconds for 100, 1.02 seconds for 200, and 7.47 seconds for 500 items. ANOVA does not show a significant difference between the two test groups ($p = 0.998$ for small and $p = 0.487$ for medium sized arrays). Although the large condition is significantly different ($p < 0.001$), this result needs to be treated cautiously due to the limited sample size after outlier removal. Nevertheless, this also confirms the previous impression that the regular binary search is the better variant of the two binary approaches.

Interpretation

As a conclusion of the raw performance, it can be stated that the plain binary search outperforms the linear search. The plain binary search also outperforms the hybrid search

for two out of three conditions. There is no clear advantage when comparing the hybrid interface and the linear search interface.

Regarding error rates, users had to correct their actions in the binary search interface 26 times in 189 samples, which equals a 13.76% of samples that had a mistake in them. This seems like an acceptable value, especially considering that there were no expert users in the tests. On the hybrid search, 36 undo and reset actions were counted, which equals a 19.05% share of trials that users had to correct.

To assess other aspects beyond raw task-completion times, we used the NASA-TLX test as final part of the study. Table 2 shows the results of the questionnaires. We focus on the *raw TLX* values as discussed by Hart [7], i.e. results are reported and analyzed separately for each category and not combined into a single TLX score. Values therefore range from -10 to 10, with lower values being better for all measures except performance.

TLX Category	Linear	Binary	Hybrid
Mental Demand	**-7.5**	2.15	3.95
Physical Demand	**-4.9**	-1.35	-0.2
Temporal Demand	1.05	**-0.45**	-0.25
Performance	**7.35**	5.0	4.15
Effort	2.05	**-1.45**	3.95
Frustration	-0.6	**-3.7**	-2.45

Table 2: Overview over the average NASA-TLX scores for each interface. Bold values indicate the best value in the respective category.

Based on another analysis of the normally-distributed data with ANOVA, we can conclude that significant differences exist across all 6 measures. Further analysis with Tukey HSD shows that pairwise significant differences for all three combinations exist for Mental Demand and Effort. For Physical Demand and Performance, the linear condition is significantly different from each of the binary/hybrid conditions, although no significance exists between the two binary search variants. Finally, for Temporal Demand and Frustration, only the difference between linear and binary search is significant, both other combinations are not.

A noteworthy result is that linear search was the only mode which users did not find to be mentally demanding, even though statistical performance evaluation showed better overall results for binary search. This also offers a possible explanation as to why the hybrid mode could not compete with the binary search: the interface and its two steps apparently become too demanding with a rating of 3.95. However, this contradicts our initial assumption that the option

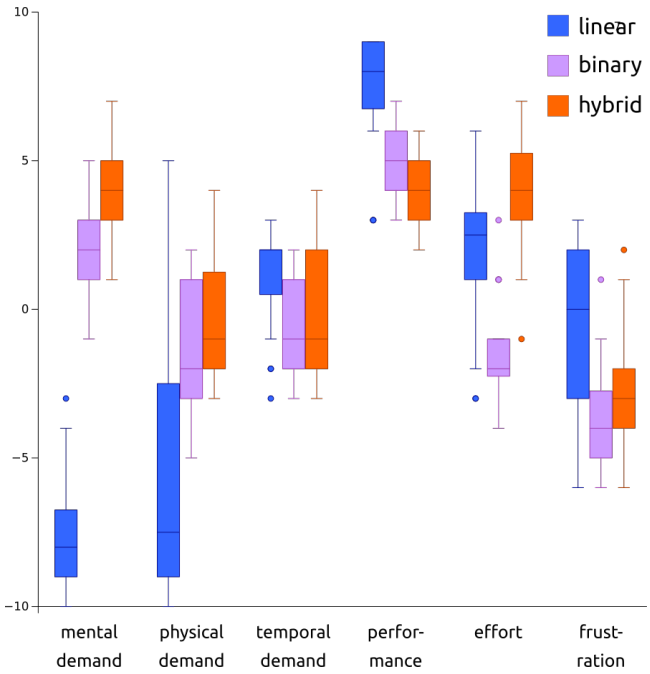

Figure 4: Nasa TLX scores for each condition and category

to choose a first letter and then just browse the respective name array would actually decrease mental demand.

Also very interesting is the disparity in performance. Users were not as confident in their performance in the two binary search options as they were in the linear search. This fact is somewhat paradox, since all users in the tests have been able to finish all their search tasks. Apparently, the test interface still doesn't provide enough clues as towards when a task is successfully finished. For real life applications, however, those clues do not make a lot of sense because the users would only find a word they specifically wanted to find and therefore also decide themselves when their task is successfully finished.

Another metric that shows results worth mentioning is the effort. Users found that the interface they had to put into the most effort was the hybrid search. Since the hybrid search is the only interface that required two different steps and also, in a way, two different searches, this makes sense. It is also remarkable that the binary search heavily outperformed the linear search as well as the hybrid search with regards to effort, being the only interface that reaches a negative value here which corresponds to the users having to put low effort into the task, which again speaks strongly for the simplistic design of the binary search application.

Temporal Demand and Frustration both rank the searches as binary being the most pleasurable, the hybrid search as the runner-up and the linear search as the weakest of the three interfaces. The results in the qualitative time category

Comparison	Mental Demand	Physical Demand	Temporal Demand	Performance	Effort	Frustration
Linear & Binary	**< 0.001**	**0.004**	**0.018**	**< 0.001**	**< 0.001**	**< 0.001**
Linear & Hybrid	**< 0.001**	**0.002**	0.058	**< 0.001**	**0.021**	0.056
Binary & Hybrid	**0.003**	0.190	0.720	0.040	**< 0.001**	0.038

Table 3: Pairwise Tukey post-hoc comparisons of all NASA-TLX Values from the User Test. Bold values show statistically significant differences (after Holm-Bonferroni correction).

and the quantitative time category from the prior subsection match. Even though users were (according to the Performance question) most confident in their results on the linear interface, they still felt frustrated by the longer searching periods and long lists.

Conclusions From the Main Evaluation

Based on our results, we consider the binary search interface to be the best-performing interface out of the three for the task at hand. The hybrid interface and the linear interface have not been able to reach the same level of performance that the binary interface produced. While the results from the NASA-TLX questionnaire were not entirely unfavourable towards the linear search interface, the binary search showed better results in two categories that are very relevant to HCI in general, effort and frustration. Even though Mental Load is still higher for the binary search, this tradeoff can be considered acceptable as long as the users do not feel overwhelmed (Effort) or become frustrated with how hard the task is (Frustration).

As mentioned earlier, the impressions from the quantitative and qualitative values match. However, the NASA-TLX shows that the mental demand for both binary and hybrid interface is considerably higher than it is for linear search. While this result is not surprising, it is important to point out that this requires further investigation for various real-life scenarios (see also following section 5).

Finally, it is worth noting that the standard deviation of the time values differs heavily between conditions. The binary search showed standard deviations of 3.09 seconds, 4.51 seconds and 4.32 seconds for the small, medium and large arrays respectively. At the same time, the hybrid search had standard deviations of 3.74 seconds, 3.52 seconds and 3.36 seconds respectively. These numbers are considerably lower compared to the linear search, that showed 5.78 seconds of deviation on the small, 6.61 seconds on the medium, and 7.90 seconds for large list size. The overall spread of completion times is higher in the linear search — simply because there are targets in the list that can be found quickly and just as many that will take a very long time to find. It could be assumed, based on these deviations, that binary search in general will provide more predictable search times - while not having many "instant find" scenarios, it also has less "long search" scenarios.

5 DISCUSSION & FUTURE WORK

The binary search interface presented in this work proves to be an option that not only outperforms the linear search, but also reaches better overall values in qualitative measures. Even though these results are convincing, the users have struggled to use the two-stage binary search interface, which was initially designed to be an improvement over the binary interface.

One limitation of our tests is the implicit assumption that the user knows that the list contains the word they are looking for. Otherwise, the user will continue to narrow down the search area where they expect the word, but will eventually limit the search range down to a point where they can be sure that they missed the word. At this point, the user can either conclude that the list does indeed not contain the word they are searching for, or could (erroneously) conclude that they made a mistake and will try to fix this through an undo or reset command, likely leading to frustration after repeated futile attempts. At the moment, this problem can only be solved through a complementary search field.

In terms of future user tests, it would be valuable to analyze whether users would actually enjoy the interface in real life. A controlled setting is important for user tests, but the validity of such a lab setting is limited to similar conditions. Using the interface in traffic, on the subway or even just during dinner for music selection are all possible and relevant scenarios that this test does not cover. There is only one realistic way to find out whether these are actually scenarios in which users, in particular experienced smartwatch users, would use the application, and this is a long-term deployment for everyday scenarios. Obviously, this requires the implementation of the interface into other standard watch applications such as the music or contact application. A long-term test would provide an additional outcome: learning how expert users interact with such an interface. Extended data gathering would also be able to show if there is a learning effect, and if so, how strong that effect is.

When looking at usage scenarios for the search interface, another issue that our present user test does not cover is the topic of cognitive load. Cognitive load generally describes mental effort that is being used to perform any task [9, 16] and is partly represented in the NASA-TLX tests under "mental demand". As stated before, the results for mental demand are quite high for both binary search approaches. While a controlled study was able to produce promising results, it still is a test that imposes little pressure and no multitasking requirements on the participants and therefore, the possibly problematic mental load does not show its effect in these studies. All scenarios named earlier in this section add some form of multitasking to the interface's use and might therefore turn the high mental load into a much more problematic issue than it appears after controlled tests.

Although the tested interfaces had a clear separation between binary and linear search, this is not a strict requirement. For example, an initial binary search step could be combined with the linear interface. This approach adds four design options; the only one of those that was fully tested in this paper, however, is the variant that uses binary search in both steps. While the implications of our user tests are that binary search generally performs better than linear search, it is quite possible that another two-step interface may still lead to better results, or even just better qualitative measures.

Dunlop et al. [5] presented a text entry interface, which used swiping gestures as confirmation and undo button. This is a very elegant way to solve the problem of lack of space on the smartwatch. As of now, our approach uses a larger button for the confirm interaction and a two smaller ones for the undo and reset commands. A possible modification would be to use a small, rounded reset button and left/right swipe for undo and confirm. This gives additional emphasis to the swiping interaction and does not require the users to switch their mental space to button interaction.

REFERENCES

[1] Sunggeun Ahn, Jaeyeon Lee, Keunwoo Park, and Geehyuk Lee. 2018. Evaluation of Edge-based Interaction on a Square Smartwatch. *Int. J. Hum.-Comput. Stud.* 109, C (Jan. 2018), 68–78. https://doi.org/10.1016/j.ijhcs.2017.08.004

[2] Frank R. Bentley and Ying-Yu Chen. 2015. The Composition and Use of Modern Mobile Phonebooks. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 2749–2758. https://doi.org/10.1145/2702123.2702182

[3] Luca Chittaro and Luca De Marco. 2005. Evaluating the Effectiveness of "Effective View Navigation" for Very Long Ordered Lists on Mobile Devices. In *Proceedings of the 2005 IFIP TC13 International Conference on Human-Computer Interaction (INTERACT'05)*. Springer-Verlag, Berlin, Heidelberg, 482–495. https://doi.org/10.1007/11555261_40

[4] Christian Corsten, Simon Voelker, Andreas Link, and Jan Borchers. 2018. Use the Force Picker, Luke: Space-Efficient Value Input on Force-Sensitive Mobile Touchscreens. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 661, 12 pages. https://doi.org/10.1145/3173574.3174235

[5] Mark D. Dunlop, Andreas Komninos, and Naveen Durga. 2014. Towards high quality text entry on smartwatches. In *CHI EA '14 CHI '14 Extended Abstracts on Human Factors in Computing Systems*. ACM, Toronto, Ontario, Canada, 2365–2370.

[6] George W. Furnas. 1997. Effective View Navigation. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '97)*. ACM, New York, NY, USA, 367–374. https://doi.org/10.1145/258549.258800

[7] Sandra G. Hart. 2006. Nasa-Task Load Index (NASA-TLX); 20 Years Later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50, 9 (2006), 904–908. https://doi.org/10.1177/154193120605000909 arXiv:https://doi.org/10.1177/154193120605000909

[8] Benjamin Lafreniere, Carl Gutwin, Andy Cockburn, and Tovi Grossman. 2016. Faster Command Selection on Touchscreen Watches. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 4663–4674. https://doi.org/10.1145/2858036.2858166

[9] Fred G. W. C. Paas and Jeroen J. G. Van Merriënboer. 1993. The Efficiency of Instructional Conditions: An Approach to Combine Mental Effort and Performance Measures. *Human Factors* 35, 4 (1993), 737–743. https://doi.org/10.1177/001872089303500412 arXiv:https://doi.org/10.1177/001872089303500412

[10] Simon T. Perrault, Eric Lecolinet, James Eagan, and Yves Guiard. 2013. Watchit: Simple Gestures and Eyes-free Interaction for Wristwatches and Bracelets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1451–1460. https://doi.org/10.1145/2470654.2466192

[11] Katrin Plaumann, Michael Müller, and Enrico Rukzio. 2016. CircularSelection: Optimizing List Selection for Smartwatches. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers (ISWC '16)*. ACM, New York, NY, USA, 128–135. https://doi.org/10.1145/2971763.2971766

[12] Philip Quinn and Andy Cockburn. 2009. Zoofing!: faster list selections with pressure-zoom-flick-scrolling. In *OZCHI '09 Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest*. ACM, Melbourne, Australia, 185–192.

[13] Anurag Rao. 2016. Automatic partitioning of a list for efficient list navigation. US Patent App. 15/224,299. Filed July 29th, 2016.

[14] Guy Shani, Chris Meek, Tim Paek, Bo Thiesson, and Gina Venolia. 2009. Searching Large Indexes On Tiny Devices: Optimizing Binary Search With Character Pinning. 257–266. https://www.microsoft.com/en-us/research/publication/searching-large-indexes-on-tiny-devices-optimizing-binary-search-with-character-pinning/

[15] Gaganpreet Singh, William Delamare, and Pourang Irani. 2018. D-SWIME: A Design Space for Smartwatch Interaction Techniques Supporting Mobility and Encumbrance. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 634, 13 pages. https://doi.org/10.1145/3173574.3174208

[16] John Sweller. 1988. Cognitive Load During Problem Solving: Effects on Learning. *Cognitive Science* 12, 2 (1988), 257–285. https://doi.org/10.1207/s15516709cog1202_4

[17] Robert Xiao, Gierad Laput, and Chris Harrison. 2014. Expanding the Input Expressivity of Smartwatches with Mechanical Pan, Twist, Tilt and Click. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 193–196. https://doi.org/10.1145/2556288.2557017