Data visualization and interaction on smartwatch small screens

by

Ali Neshati

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
for the doctoral program
under the supervision of Dr. Pourang Irani

Department of Computer Science
University of Manitoba
Winnipeg, Manitoba

Dec 2021

ABSTRACT

In this thesis, I will investigate the optimization of a small display while presenting graphic data such as line charts. Due to the small screen of smartwatches presenting the data could be challenging. To overcome these challenges, I will propose two techniques, Compression and Simplification, to improve the visualization techniques on smartwatches. The last part of this thesis is about interaction with visualization techniques on smartwatches. I will show how interacting with the smartwatch bezel and using some specific parts of the smartwatch display can be helpful to interact with various types of charts and graphs on smartwatches.

In the first part of my thesis, I propose G-Sparks, a compact visual representation of glanceable line graphs for smartwatches. My exploration primarily considered the suitable compression axes for time-series charts. In a first study, I examine the optimal line-graph compression approach without compromising perceptual metrics, such as slope or height detection. I evaluated compressions of line segments, the elementary unit of a line graph, along the x-axis, y-axis, and xy-axes. Contrary to intuition, I find that condensing graphs yield more accurate reading of height estimations than non-compressed graphs, but only when these are compressed along the x-axis. Building from this result, I study the effect of an x-axis compression on users' ability to perform "glanceable" analytic tasks with actual data. Glanceable tasks include quick perceptual judgements of graph properties. Using biometric data (heart rate), I find that shrinking a line graph to the point of representing one data sample per pixel does not compromise legibility.

As expected, such type of compression also has the effect of minimizing the needed amount of flicking to interact with such graphs. From the results, I offer guidelines to application designers needing to integrate line charts into smartwatch apps.

Multiple embedded sensors enable smartwatch apps to amass large amounts of interrelated time-series data simultaneously, such as heart rate, oxygen levels or steps walked. Visualizing multiple interlinked datasets is possible on smartphones but remains challenging on small smartwatch displays. I propose a new technique, the Space-Filling Line Graph (SF-LG), that preserves the key visual properties of time-series graphs while making available space on the display to augment such graphs with additional information. Results from the first study (N=30) suggest that, while SF-LG makes available additional space on the small display, it also enables effective (i.e. quick and accurate) comprehension of key line graph tasks. I next implement a greedy algorithm to embed auxiliary information in the most suitable regions on the display. In a second study (N=27), I find that participants are efficient at locating and linking interrelated content using SF-LG in comparison to two baselines approaches. I conclude with guidelines for smartwatch space maximization for visual displays.

I present BezelGlide, a novel suite of bezel interaction techniques designed to minimize screen occlusion and 'fat finger' effects when interacting with common graphs on smartwatches. To explore the design of BezelGlide, I conducted two user studies. First, I quantified the amount of screen occlusion experienced when interacting with the smartwatch bezel. Next, I designed two techniques that involve gliding the finger along the smartwatch bezel for graph interaction. Full BezelGlide (FBG) and Partial BezelGlide (PBG) use the full or a portion of the bezel, respectively, to reduce screen occlusion while scanning a line chart for data. In the common value detection task, I find that PBG outperforms FBG and

Shift, a touchscreen occlusion-free technique, both quantitatively and subjectively, also while mobile. I finally illustrate the generalizability potential of PBG to interact with common graph types making it a valuable interaction technique for smartwatch users.

As the last piece of this thesis, I will show how some specific segments of the smartwatch display can be used as the interaction area to interact with multiple graphs on smartwatches. In the first experiment, I will investigate the occlusion area while the smartwatch user interacts with the whole smartwatch display. The result of this study will help to pick the right segments with minimum occlusion to interact with multiple graphs. For the interaction technique, the challenge will be mapping the selected segments (interaction area) with the area representing content.

The key findings of this thesis can be summarized as follow:

- Line graphs can be compressed for legibility on small screen of smartwatches.
- Complex graphs can be simplified by deploying space-efficient algorithms, to create new space for showing auxiliary data.
- Interaction on the bezel is possible for exploring graphs on small screens of smartwatches, mitigating the screen occlusion and fat finger problems with small displays.
- Interacting on the entire watch face requires careful consideration of finger occlusion.

PUBLICATIONS

This report is an original work by the author. Contents presented in Chapter 3-5 of this report have already been published in premier venues in Human-Computer Interaction.

1. Neshati A, Sakamoto Y, Leboe-Mcgowan L, Leboe-McGowan J, Serrano M, Irani P. G-sparks: Glanceable sparklines on smartwatches. In45th Conference on Graphics Interface (GI 2019) 2019 May 28 (pp. 1-9).

2. Neshati A, Sakamoto Y, Irani P. Challenges in Displaying Health Data on Small Smartwatch Screens. InITCH 2019 Jan 1 (pp. 325-332).

3. Neshati A, Rey B, Ahmed Sharif F, Bardot S, Latulipe C, Irani P. BezelGlide: Interacting with Graphs on Smartwatches with Minimal Screen Occlusion. InProceedings of the 2021 CHI Conference on Human Factors in Computing Systems 2021 May 17.

4. Neshati A, Alallah F, Rey B, Sakamoto Y, Serano M, Irani P. SF-LG: Space-Filling Line Graphs for Visualizing Interrelated Time-series Data on Smartwatches. submitted to In23rd International Conference on Human-Computer Interaction with Mobile Devices and Services 2021 Oct 5.

FOREWORD

First and foremost, I would like to give my biggest thanks to Dr. Pourang Irani for being an amazing mentor and advisors both personally and professionally throughout this adventure. I want to thank Dr. Dan Vogel, Dr. Lisa lix, and Dr. Mohammad Noman for being on my committee. I thank my fellow colleagues in the HCI lab for their support and guidance throughout this experience. I also would like to thank my wife, Maryam, my daughter, Nicki, and my parents for their unconditional support.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Wearable devices such as smartwatches and fitness bands are becoming increasingly popular across demographic groups, from kids [6, 65, 117] to older adults [27, 57]. Reasons for this significant uptake include fitness tracking [11, 124] and health monitoring capabilities [13, 52, 70, 76, 85], two of the most widely available features on a smartwatch. Other health-related capabilities of smartwatches such as detecting mental/physical disorders and supporting people with difficulties [9, 67, 107] can also improve user health significantly.

Being lightweight, accessible, and having various sensors and a wide range of novel interaction techniques make smartwatches unique and ubiquitous as a data-tracking device. A wide range of embedded sensors on smartwatches provides users with various personal time-series data, such as heart rate, breathing rate, oxygen levels and electrodermal activities. Smartwatch users are becoming increasingly dependent on such personal data to adjust their activity levels and behaviour [28], also referred to as in-situ analytic [22].

Before the emergence of smartphones, personal computers were the dominant digital devices people were using for their day-to-day tasks for many years. After the evolution of smartphones, we realized that many of these tasks could be done by a device that is in everyone's pocket, even though they have smaller screens and less computational power. Many researchers created new novel interfaces and

creative interaction techniques to replace personal computers with smartphones for many of the tasks we were using our personal computers. This means that these days, for many of our daily tasks, using a personal computer is not as convenient as using a smartphone, and it is not necessary. Using a device that is on our wrist all the time is even more convenient than using a smartphone. If there are ways that we can use our smartwatches to perform the same tasks we were using our smartphones or personal computers, it will have a significant effect on people's lives.

1.1 DATA VISUALIZATION

Data visualization is a powerful tool that makes the collected data more insightful to the users. The raw data gathered by smartwatches can be visualized and analyzed in such a way that users can extract useful information about their daily activities, such as steps taken, average/rest heart rate, intake/burnt calories. This data can also be leveraged to motivate habitual changes for health purposes,such as the cessation of smoking [3]. When the data is presented and interpreted properly, it can also help professional athletes to improve their performances [129]. Many software, such as Tableau and Microsoft Power BI, are specifically designed to visualize the data on computers. However, presenting the data on small displays of smartwatches could be more challenging compared to data visualization on larger displays (e.g., smartphones) due to existing limitations of smartwatches such as limited available space and computational capabilities. Also, recent work suggests that not all existing visualizations on smartwatch small displays are equally effective [12], and thus need to be designed carefully.

The small display of smartwatches is known as one of the most well-known barriers to represent information. Visualizing health data, which are mostly highly dense, continuous time-series data, is often challenged by these small screen sizes. Therefore, novel visual representations are needed to maximize the available screen real-estate, not only to show key visual properties but also to include data from complementary data sources. Furthermore, often needed on-the-go queries can benefit from being visualized on smartwatches, which are not currently supported [5].

As depicted in Figure 1.1, One of the main goals of this thesis is to investigate the common visualization techniques that exist on smartwatches (e.g., line graphs) and to enhance these techniques based on smartwatch characteristics (e.g., limited available space to visualize the data and interact with it). Our proposed techniques are designed to overcome the limited available space on smartwatch displays and to improve the effectiveness of standard visualizations to the smartwatch users. To achieve our goal, we will propose two different approaches to visualize the data on such small displays.

1.1.1 *Compression*

In this technique, we ask how best to shrink line graphs, as one of the most commonly used visualization techniques to represent time-series data, to minimize interaction effort without sacrificing glanceability on smartwatches. Inspired by Sparklines [108], a technique to compress line graphs on the y-axis (height of the line graph) and integrate it like words in texts, images, and tables, we investigate three different ways of condensing high-density continuous time-series

data on smartwatches. Although Sparklines can be used directly on smartwatches, this technique compresses the graph mainly on the y-axis to integrate it within words. However, on smartwatches, we postulate that condensing the graphs on smartwatches, such that they are legible, can enable a host of new scenarios. For instance, showing the user's continuous heart rate history, besides other information such as sleep quality and breathing patterns, can better inform the user of their stress level and make them aware of their biometric states. A key design challenge we address is a method to see a large amount of the dataset on the smartwatch by condensing the graphs without compromising the glanceability aspects necessary for in-situ tasks. We explored the dimensions (x-, y-, or both) best suited at shrinking a line graph while still being able to respond to basic perceptual and judgement tasks.

1.1.2 *Simplification*

Unlike smartphones which can present multiple interlinked datasets, the small smartwatch displays make relating different data points difficult, a feature needed for performing tasks such as "am I running faster than I did yesterday?" or "how did the change in my walking speed affect my heart rate", among others. In this technique, we explore means to effectively use a small display of a smartwatch for presenting interlinked content. We consider how best to segment and fill content around a line graph in order to facilitate users' complex time-series data exploration. We introduce the Space-Filling Line Graph (SF-LG), which is designed first to simplify a line graph to make available additional screen space and sec-

ond to use the created space to place additional visuals to aid with complex queries.

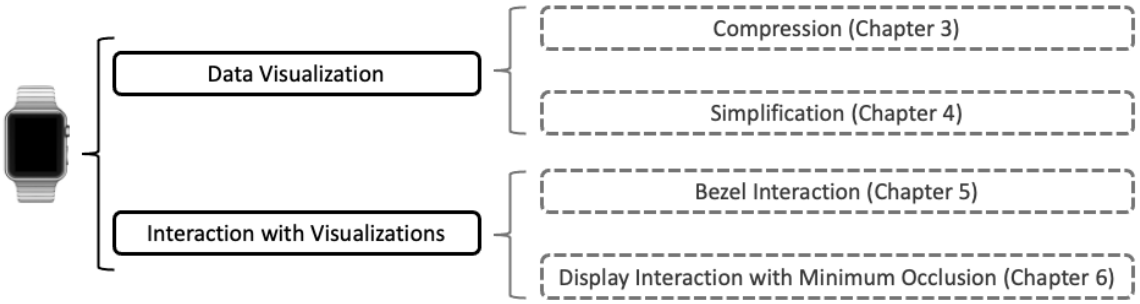


Figure 1.1: The two major topics in this thesis are data visualization and interaction with them on small display of smartwatches.

1.2 INTERACTION WITH DATA VISUALIZATION

Our visualization techniques are designed to provide an overview of often complex collected data by reducing details or simplifying the visual presentation. This simplification of data is essential for smaller screen displays, such as those found on smartwatches. However, while research efforts have often focused on designing the visualizations appropriately, the loss of detail in the data does not allow for full data exploration when needed. Often times, exploration of collected data from a smartwatch requires using an auxiliary smartphone to fully explore the data. With the benefits that smartwatches provide, due to their always accessible nature, improved data visualization interaction techniques could allow users to fully explore their data collected while in-situ and directly on the watch.

Generally, designing any interaction technique for smartwatches can be challenging due to the limited display [78] and input space [7]. Even common techniques used on smart devices, such as tapping and flicking, can not be performed as

effectively on smartwatches as they are on smartphones [103]. In addition, direct interaction with the content on a small display of smartwatches may block the content because of the big size of the finger compared to the size of the content on the smartwatch display, known as the 'fat finger' problem, making the interaction a challenging task. Even indirect interaction with the smartwatch screen (e.g., smartwatch bezel) can occlude some segments of the smartwatch display from the smartwatch users. When interacting with data visualizations on the watch, this screen occlusion can block a user's view of many necessary display elements, making the interaction ineffective. Researchers have proposed novel interaction techniques to expand the limited smartwatch input modalities [40, 56, 80, 115, 126]. However, such interactive capabilities were not specifically designed for the common task of exploring data on smartwatches, including line-, bar-, donut-, and radial bar-charts.

The second major goal of this thesis is to propose interaction techniques specifically designed to interact with multiple visualization techniques on smartwatches for further data exploration. These techniques should be able to overcome the Fat finger and screen occlusion problems on smartwatches.

1.2.1 *Bezel Interaction*

To mitigate the 'fat finger' and screen occlusion problems on smartwatch displays, we explore the use of the bezel to facilitate exploration. This focuses the interaction away from the center of the screen where the visualization is viewed, and utilizes often untouched pixels on smart device displays [91]. We postulate that such a mechanism may mitigate the 'fat finger' and screen occlusion effects, albeit

providing only indirect input. While certain smartwatches allow physically rotating bezels, soft bezel solutions have shown to improve smartwatch input [59, 87, 123], particularly when the device is circular, the form-factor we focus on for our exploration.

1.2.2 *Interaction with Smartwatch Display with Minimal Occlusion*

Smartwatch display is the primary input modality of a smartwatch. However, screen occlusion is one of the main issues reported by previous research to prevent users from appropriately interacting with the smartwatch display content. The proposed bezel interaction technique is designed to overcome the screen occlusion issue. However, it has some limitations. For instance, the screen occlusion exploration was done only on the smartwatch bezel and not the entire display. Also, the interaction technique was not designed to interact with multiple graphs which is a common technique to visualize interrelated data. In addition, the nature of our bezel interaction technique is a linear interaction approach which is not appropriate to interact with multiple graphs. Therefore, this section of this thesis will generalize our bezel interaction technique in different ways. We will explore the screen occlusion issue with the entire smartwatch display. The result of this investigation will provide us information about segments of the smartwatch display with minimal screen occlusion that can be utilized for interaction with the smartwatch screen. We show that our non-linear interaction technique can be used to interact with various types of graphs.

Based on what was discussed, here are the main objectives of this thesis:

- **Obj1:** Can we enhance existing visualization techniques on smartwatches?

- **Obj2:** How do smartwatch visualizations optimize the display space?
- **Obj3** Can multiple data sources be represented in a space efficient manner?
- **Obj4** How can interaction methods be designed to support multiple forms of visualizations?

2 RELATED WORK

The related work chapter can be divided into two main sections. The first section is about the data visualization techniques. The second section is about state-of-the-art interaction techniques on smartwatches. These papers inspired us to propose our visualization techniques and interaction methods, designed for smartwatches.

2.1 DATA VISUALIZATION

We provide an overview of relevant work on small screen visualization techniques and graphical perception in this section.

2.1.1 *Information Visualization on Small Screens*

Consumer interest in monitoring continuous human activities, which can be achieved by an unobtrusive device such as a smartwatch, is increasing rapidly [71]. When the data is interpreted properly, it can enrich our lives, particularly when it concerns health-related and personal activities (e.g., How many calories did you use today?) [88]. Li et al. [62] identified six major questions that consumers regularly ask when looking at their personal data. These center around Status, History, Goals, Discrepancies, Context, and Factors. These center around Status,

History, Goals, Discrepancies, Context, and Factors. Questions related to the consumer's current condition belong to the Status category. For instance, what is the current heart rate or stress level of the smartwatch user? Questions related to personal insights which can be gained from the collected data over a period of time, belong to the History category. This means the user needs to investigate and analyse the collected data over a specific period to answer these questions. In addition, many of the users' queries are related to their goals. In this case, the user defines new goals by investigating their data. For example, the user checks the screen time collected by their smartphone and might see that the amount of time they spend on their cellphone is significant. In this case, the user's goal is to minimize screen time. The discrepancy category is about all the questions related to comparing the user's current status with their goal. This will help the user to adjust their behaviour accordingly. For instance, a runner may increase or decrease their current speed to maintain a specific heart rate range. Context and Factors categories are similar. In the Context category, the user investigates how multiple data sources may affect their current status. For instance, how blood sugar and stress level can affect their current mood. In contrast, in the Factors category, questions are about investigating the effect of multiple factors on consumer behaviour over a long period of time. For example, how losing weight and exercise can affect the general wellbeing of the user. Effective data visualizations are essential to find answers to these six major questions on smartwatches throughout the day [47].

There exist a number of works concerning the representation of health data, such as for introducing novel analytic and visualization tools for Electronic Health Records (EHR) [122]. For example, Wang et al. [116] introduced an interactive visual tool that helps professionals to detect hidden patterns in patients' data,

which otherwise can be hard to detect. Health data is usually complex massive time-series data. For instance, heart rate and Electrocardiogram or ECG, respiratory data, and body temperature are time-series data. Many health data types are crucial and it is crucial for physicians to have real-time access to such data and visualizations [46]. Dimension reduction techniques have been used to make complex massive health data simpler and easy to understand. For instance, the Perceptually Important Point (PIP) algorithm is a simplification approach destined to reduce the complexity of a line graph by selecting and representing the essential data points of the graph rather than representing all data points. Fu et al. [30] showed how PIP can be applied on line graphs, as the most common way to represent large ECG data. In this paper, the authors showed that PIP can select just a small portion of data points in a line graph representing ECG and still maintain the general trend and essential points of ECG.

Line graphs are highly common as they convey many aspects of temporal data. Novel line graph designs have long been investigated by many researchers [50, 83]. Sparklines, proposed by Tufte [108], compresses line graphs, usually without axes or coordinates, to represent high resolution and continuous time-series data. Sparklines is designed to be embedded in texts, images, and tables, with small fonts, and to provide a high-level overview of the data. Sparklines compress line graphs usually on the y-axis such that it fits within the font size selected for its associated text. However, designers of line graphs on smartwatches are not necessarily constrained by font size, and can compress line graphs on the x-, y- or xy-axes. Furthermore, when compressed, line graphs on smartwatches can be strategically positioned to convey the necessary information.

A few studies explore visualizations on small screens. For instance, Blascheck et al. [12] investigated how quickly users compare two data points, using three

different visualization methods (e.g., bar chart, donut chart, and radial bar chart) and three different sizes of data sets (e.g., 7, 12, and 24 data values), on small smartwatch screens. In two similar studies, they showed participants different charts with two indicated data points in each chart, and participants were asked to determine which one of these two data points has higher value. They collected response times of participants to determine which one of three chart types is the most glanceable chart for data comparison task. In this paper, the authors showed that the radial bar graph was the least glanceable visualization technique compared to a bar graph and donut chart. This indicates that even standard existing visualization techniques used on smartwatches might not be effective in practice.

Fulk [33] shows that we should take specific approaches to represent web content on small displays. Split View and Fisheye techniques can help users focus only on a specific part of a webpage. Minimap also allows presenting selected information on a small display [92]. The RSVP Browser provides users with brief information of all links of web pages as images, to accommodate mobile device screens [25]. Wedge is another technique that shows off-screen content in an accurate manner, but on the smartphone display [38]. Similarly, EdgeRadar allows the compressed visualization of off-screen moving items on the smartphone display [37].

Researchers have further explored techniques to resolve the limited display issue at application specific levels. For instance, with filtering techniques, representing a reasonable amount of details specifically by regulating the number of relevant elements in the map, we can have a better understanding of geo-referenced information on small screens [16]. Similarly, in [82], researchers noticed that designing an interface for small screens to visualize electronic health records can be a challenging task, primarily because of the large volume of records.

An alternative to the above involves modifying the small display itself [69] with additional hardware. For instance, in [97], researchers combined the output of a head-worn display and smartwatch screen to provide additional information. Likewise, Facet is a multi-display wrist-worn smartwatch, containing an array of multiple touch-sensitive segments [66]. Further, using a similar approach, Wenig et al. [121] deployed an additional transparent display beside the original screen to represent more information on it.

Smart approaches can also optimize the available space. Techniques such as SpiraList and SnaiList help display long lists on small screens using a spiral layout [48]. Similarly, techniques such as Smartfonts and SmartRSVP can facilitate reading small texts on small wearable devices [14, 35]. In [14] researchers introduced a new series of fonts that are appropriate to read on small screens, and in [35] they investigated the application of "Rapid Serial Visualization Presentation" which represents words of text separately with adjustable speed.

It is worth pointing out that our literature search yielded only a handful studies addressing the issue of information visualization on smartwatches [5, 12, 49]. Further, we did not find any clear guidelines or research focusing on the size nor the compression methods of line graphs on smartwatches. For instance, how compressing a line graph on the different axis may affect the perception and performance of the smartwatch users understanding the compressed graph. Such guidelines will help the smartwatch application designers properly adjust the graphs' size to represent the data.

2.1.2 *Graphical Perception*

Graphical perception is defined as "the visual decoding of categorical and quantitative information from a graph" [20]. Graphical perception hinges on the graph size, area, color, angle, position, length, and other various factors. It is important to explore how individuals interpret different types of graphic representation techniques. For instance, authors in [58] introduced a perceptual guideline for treemap charts.

Numerous studies investigated graphical perception differences based on different graph types [23, 24, 26, 50, 84]. Further, Cleveland et al. [19] examined ten elementary perceptual tasks people go through (e.g., shading, angle, length, direction, area, and position) when extracting quantitative information from certain types of graphs. Similarly, Cleveland et al. [20] investigated and discovered the effect of six factors such as length, angle, slope, and area, on extracting information from graphs. We incorporated some of these factors in our study.

Several studies exploring users' graphical perception with varying graph types [23, 24, 26, 50, 84] identified five essential perceptual tasks. These include (i) maximum/minimum point detection [4, 29, 50, 60, 64, 83], which requires finding the highest or the lowest point in a graph (e.g., peak daily heart rate); (ii) value detection, which requires reading an exact data point [45, 50, 60] (e.g., comparing peak heart rate on different days); (iii) value comparison [1, 12, 29, 50, 74, 102], which requires comparing two data points and identifying which one is higher, or how much is the difference between two highlighted data points in a chart; (iv) trend detection [32, 120], which can give users an overview of the entire data; and

(v) slope detection and slope comparison. [10, 50, 83]. Thus, from the findings in these works we incorporate these perceptual tasks in our evaluations.

2.1.3 *Smartwatch data visualization*

For smartwatches, techniques for accurately collecting data have outpaced those for visualizing data. Recent studies have suggested how visuals on such small displays could assist with complex tasks [28, 36, 124]. Through focus group studies, Amini et al. [5] identified some of the common queries end-users would desire to accomplish on smartwatches. They concluded that many of the tasks users prefer accessing while on-the-go are not supported by current platforms. With the aid of visual designers, they propose a range of visualizations, many of which have yet to be designed.

In an elaborate evaluation, Blascheck et al. [12] highlight the need to carefully choose visualizations on smartwatches. Their investigation compared the three most commonly used data charts (bar, donut, and radial bar charts), with a task requiring participants to quickly compare two data point values. Interestingly, their results suggest that the radial bar chart, one of the most common visualizations on smartwatches, is the least efficient for quickly viewing data on smartwatches. Furthermore, we note that line graphs can also be compressed to show more content. G-Sparks [74], a method inspired by Sparklines [108], compresses a line graph to include one data sample per pixel which can minimize the number of flicks required to access lengthy time-series data on smartwatches. These prior works point at the need for renewed guidelines to address the growing interest towards data visualizations on small form-factor devices.

Please note that although Blascheck et al. [12] showed that bar, donut and radial bar charts are the most common graph types on smartwatches, the focus of this thesis is mainly on line charts for a few important reasons. The most important reason is that most of the collected data on smartwatches are time-series data, and one of the most common ways to represent time series data is a line graph. The second reason is that they only investigated Android smartwatch applications in their paper. So we did a similar investigation both on Android and Apple platforms, and we found that line graphs are one of the three common data visualization techniques used on smartwatches (check Chapter 4 for more details). Although bar chart was reported as the most common existing visualization technique on smartwatches, both in [12, 72], it is not a space-efficient visualization technique (using height and area of bars to represent the data). Line charts and bar charts are very similar, so that line graphs can be used as a more space-efficient alternative visualization technique which is essential on small displays of smartwatches.

2.1.4 *Space-efficient data presentation*

Space-efficient visualization techniques take up a minimal footprint [74, 83, 95, 108] and can be referred to as micro- or small-scale visualizations [12, 81]. The Horizon Graph [83, 95], for example, colour codes line graph components making it possible to compress these along the y-axis. By conducting a user experiment, Javed et al [50], showed how using the Horizon graph can be used for some specific visual exploration tasks. However, the colour scheme requires interpretation which can be challenging to do on a small display. Space-filling techniques are also efficient, and they are designed to maximize the available display space [51, 105,

109, 119]. As a result, space-filling techniques reduce chart junk [50] and rely on highlighting salient content [118]. The Treemap [51, 105, 118] is a classic example of a space-filling technique for hierarchical data. We build on such concepts to maximize the space around a line graph on a small display.

2.1.5 *Line graph simplification methods*

Line graph simplification, or smoothing methods, focus on decreasing the complexity of graphs by reducing the number of data points. Many such techniques exist [94, 106]. In a research paper, Rosen et al. [90] evaluated 12 different smoothing techniques and proposed a taxonomy of these techniques. However, these smoothing methods are complex and hard to implement on smartwatches with limited processing resources. In this study, we focus on three existing line graph simplification methods that inspired our design. These three simplification algorithms were chosen because they do not require powerful computational resources, making them suitable for smartwatches with limited processing power.

2.1.5.1 *Sampling:*

Sampling is one of the most commonly used simplification techniques [8] whereby fixed-size intervals are used to select data points (e.g., every minute). The larger interval sizes reduce the complexity of data by having fewer data points (e.g., every hour) and can be beneficial for trend and pattern recognition tasks. In contrast, smaller fixed-intervals generate more data points (e.g., every 5 seconds) and can be beneficial for identifying incidents. Sampling does not prioritize the data in the

way our visual system does, and thus we can potentially lose salient information (e.g., maximum and minimum) unless they coincide with the selected intervals.

2.1.5.2 *Piecewise Aggregate Approximation:*

The Piecewise Aggregate Approximation (PAA) technique simplifies line graphs using a fixed-size window. In PAA, the mean of the data points in each window is calculated, and simplification replaces the original data points with the mean of each window [54]. Thus, similar to Sampling, simplified graphs with PAA often do not contain the actual critical data points (e.g., maximum and minimum). Furthermore, the effectiveness of this method hinges on the size and the number of windows. For example, the simplified graph might not hold the shape of the original graph when the window size is large.

2.1.5.3 *Perceptual Important Points:*

Chung et al. [18] introduced the Perceptual Important Points (PIP) technique which inspired SF-LG. To compute the PIP graph, first, the furthest data point on the line connecting the first and the last points in the graph is detected. The process is then repeated for each one of the two components separately (e.g., selected data point with the first data point and selected data point with the last data point). While PIP requires intensive resources compared to Sampling or PAA, it can preserve the maximum and minimum data points. PIP was explicitly designed to reduce the complexity of line graphs with a large number of data points. It has many applications in areas such as health and stock market forecasting [30]. It is also one of the only simplification techniques that considers the visual aspect of important features in the graph during graph simplification.

2.2 SMARTWATCH INTERACTION

In this section, we summarize relevant work regarding smartwatch usage scenarios, data visualizations for smartwatches, mitigation of screen occlusion, and smartwatch interaction, including bezel interaction. We then describe how our work builds on this previous research.

2.2.1 *Smartwatch Usage*

Due to the always accessible nature of smartwatches, their use is often quick and has been shown to occur in under 5 seconds [110]. Further, smartwatches have seen increased use while on-the-go [86]. Such on-the-go use can be seen when walking to a destination, while on a transit system, or performing other tasks. The common use cases involve notifications, checking time, communication, and inspecting health data among others [17, 34, 68]. Specifically, health data, which often uses data visualizations to convey information, was utilized for two types of interaction. These two types are peeking, as well as physically interacting with the health data which occupied 57.3% of the type of interaction [110]. Due to these common usage scenarios and knowledge regarding health data interaction, as well as the understanding that mobility can affect touch interaction [75, 103], we look to explore our data visualization techniques in not only static conditions but also while walking. This will offer an improved understanding of any effects that may occur due to mobility, and justify the use of our BezelGlide techniques across usage scenarios.

To assess the legibility of such visualizations, researchers resort to a set of canonical data exploration tasks [12, 23, 26, 50, 74, 84]. For example, data value detection, in which the user is exploring the exact values of data points, is one of the most common visual queries in several studies [45, 50, 60]. Data point comparison is another task [12, 74], in which the user must look for the exact differences between two or multiple data points; or the user has to identify which of two data points has the higher or lower value. Since this task involves the comparison of values at two or more data points, it can be appraised as a more onerous version of the single value detection task. Trend detection and pattern recognition are two other commonly used tasks across different studies examining chart legibility [32, 42, 55, 63, 120]. Users perform these two tasks for seeking particular patterns and trends in charts. Such tasks are more closely associated with the specific data being represented and the ability to perform these can largely depend on the visualization used. Given that value detection is fundamental to all of these data exploration tasks, we adopt it as the primary task in our work.

2.2.2 *'Fat' Finger and Screen Occlusion Considerations During Interaction*

The limited input and output space on smartwatches results in the display being largely occluded when interaction takes place [2, 96, 104, 131]. Two main problems arise. First, the 'fat finger' effect, is a result of the width of the fingertip blocking the exact touch location and contents underneath [101]. For example, selecting a point on a graph results in occluding both that segment of the graph, but also the value that would commonly be displayed aside it. Second, screen occlusion occurs when the interaction blocks content of interest on other areas of the screen due

to the body of the finger across the screen. This may force the user to move their head or adjust the interacting point to better examine the display. This can occur on a smartwatch, when a right handed finger is interacting with the upper left corner of a smartwatch display on the left wrist, blocking the content on the upper left corner of the display [125].

Due to these effects seen when interacting with smart displays, Vogel and Balakrishnan explored screen occlusion from the hand when interacting on a a large touch display [111]. Because of the issues that arise from these problems many interaction techniques and adaptive methods [41, 56, 96, 112, 125] have been created to overcome these issues. While many of these techniques mitigate the 'fat finger' and screen occlusion problems, on smartwatches they often utilize external hardware or are input modalities that are not common to users. Further, these techniques are not designed to interact with data visualizations, rather focusing on smartwatch interface navigation and control.

Shift, as an input technique, was designed such that content directly under the fingertip moves, or shifts away for better legibility [112]. Shift is highly effective in a number of settings, including on smartwatches [61, 99], but also in comparison to techniques such as ThumbSpace[53], TapTap or MagStick [93].

What is unknown in research, is the degree to which interaction with the bezel of a smartwatch occludes the smartwatch screen itself. Measuring the degree of screen occlusion can help further determine the optimum positioning of information on the display, and to provide more suitable interaction techniques. Moreover, these interaction techniques must also aim to provide control of visualizing data represented on the smartwatch.

2.2.3 *Bezel Interaction Techniques*

One such interaction method that has been explored on smartwatches, is to utilize the bezels. These bezels can be used for multiple forms of interaction. This includes rotary movement [15, 128], bezel taps [130], bezel initiated sequential tapping [98], bezel pressing [127], edge based interaction [2, 41, 87], bezel initiated swiping [41, 123], and bezel to bezel [59]. Of note, is edge based interaction, as seen in [2], which gives the user an indirect form of interaction with content, as the point of interaction is at the edge, while the actual display may be in the screen centre. However, 'square watch interaction' (SWI) techniques proposed by [2] are specifically designed for smartwatches with square displays and require additional sensors. Although SWI techniques were designed to cover various tasks, they are not specifically designed for visual exploration on smartwatches. Furthermore, edge plus screen interaction proposed by [2], with which the user needs to interact directly on the screen, can block a considerable portion of the display.

Similarly, in 'EdgeTouch' [77], an array of touch sensors were used to detect around-the-display touch points. Since the accuracy and performance of this interaction technique relies heavily on the resolution of the array of sensors around the display, using EdgeTouch to interact with dense charts and graphs on a smartwatch display could be challenging. Moreover, if the offset of the interaction area is considerable in EdgeTouch, this may affect the user's performance.

Interestingly, smartwatch bezels have been used in commercial smartwatch applications[1] and on some devices is embedded in the hardware[2]. Even without embedded hardware, bezel techniques only need to consume an eighth of the

1 https://www.samsung.com/global/galaxy/galaxy-watch-active2/
2 https://www.samsung.com/us/mobile/wearables/smartwatches/samsung-gear-s3-frontier-sm-r760ndaaxar/

width of the screen [123], pixels which are often unused [91]. This is an important fact, in that bezel interaction may allow for data visualizations to occupy the majority of the small screen display on smartwatches while still providing a means of interaction.

Many interaction techniques have been created for smartwatches utilizing the whole touch screen, the bezel, and external hardware. While these techniques improve the navigation of smartwatch interfaces, they do not explore their use for interacting with data visualizations. This interaction is crucial, as smartwatches are continuously collecting data and providing feedback for quick and in-situ exploration. Please note that for the rest of this thesis, "small displays" refers to small display of smartwatches, which are about 30-40 millimeters in size [12].

3 G-SPARKS: GLANCEABLE SPARKLINES ON SMARTWATCHES

Line graph is one of the most commonly used type of graphs on smartwatches to present time series data collected by smartwatches (Figure 3.1). In this section we will investigate how compressing line graphs can improve the perception of smartwatch users of line graphs.

Figure 3.1: Line graphs are one of the most common data visualization techniques on smartwatches to represent large time-series data collected by sensors.

3.1 COMPRESSING LINE GRAPHS

Line graph compression involves shrinking one or two dimensions of the graph, on the x-, y- or xy-axes. While various compression techniques are possible, we explored three specific approaches.

X-Axis Compression: in this approach we compress the baseline graph only along on the x-axis (e.g., 25% compression; X:Y = 0.25:1). This can be captured by the following relationships:

$$X_{lc} = X_{lb} \times (CL)$$
$$Y_{lc} = Y_{lb}$$

where Xlc and Ylc represented the length of x-axis and y-axis of the compressed line graph, Xlb and Ylb represent the length of the x-axis and y-axis of the baseline line graph and CL represents the Compression Level (e.g. 25

Y-Axis Compression: in this method we compress the baseline graph only on the y-axis (e.g., 25% compression; X:Y =1:0.25), with the following relationship:

$$X_{lc} = X_{lb}$$
$$Y_{lc} = Y_{lb} \times (CL)$$

XY-Axis Compression: we combine the above two to compress on both x and y axes simultaneously (e.g., 25% xy-axis compression; X:Y = 0.25:0.25) (Figure 3.2). This is represented as:

$$X_{lc} = X_{lb} \times (CL)$$
$$Y_{lc} = Y_{lb} \times (CL)$$

3.2 SUTDY1: COMPRESSION TYPES

The goal of Study 1 is to identify which *Compression Type* provides the best perceptual legibility. We additionally explore the approximate compression threshold suitable in terms of pixel density for a smartwatch display.

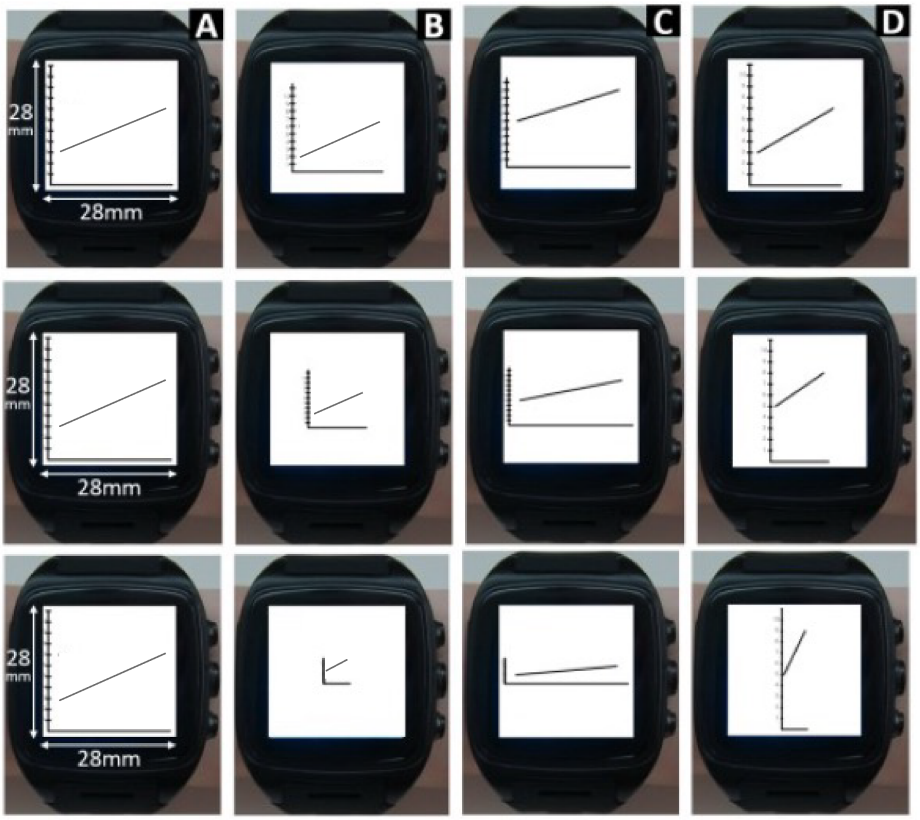


Figure 3.2: Screenshot of graphs used in Study 1 representing, a) baseline graphs, b) xy-axis compression, c) y-axis compression and d) x-axis compression technique.

3.2.1 *Graphical Perceptual Tasks*

Participants engaged in simple graphical perceptual tasks where they viewed a series of simplified line segments on a smartwatch (Figure 3.2). We asked them to identify height differences as well as the slope type (i.e., increasing or decreasing) of this most basic unit, a line segment.

Height difference:

Participants identified the absolute height difference between the start and end of a line segment. The difference could be either 0, 1, 2, 3, 4. The length of one unit on the y-axis was 20pxs for the baseline graph and all x-axis compressed graph, 10pxs for 50% of compression for xy-axis and y-axis *Compression Type*; and 5pxs for 25% of compression for xy-axis and y-axis *Compression Type*. Please note that we focused on smaller height differences (i.e., between 1 and 4) as these differences were more challenging than larger height differences (i.e., 5 to 10). Further, including all the possible height differences (i.e., 0 to 10) would necessarily increase the number of the segments drastically, which would induce cognitive fatigue. Given the relationships above, Table 1 describes the various compression amounts for the pixel density of a line segment for height 4 on our device (see Apparatus section).

	X-Axis-Length	**Y-Axis-Length**	**XY-Axis-Length**
Baseline	184 pixels,2.13 cms	68 pixels, 0.8 cms	196 pixels, 2.27 cms
xy-axis, 25%	46 pixels, 0.53 cms	17 pixels, 0.2 cms	48 pixels, 0.56 cms
xy-axis, 50%	92 pixels, 1.06 cms	34 pixels, 0.4 cms	98 pixels, 1.14 cms
xy-axis, 75%	138 pixels, 1.6 cms	51 pixels, 0.6 cms	147 pixels, 1.7 cms

Table 1: xy-axis compression method relative to the baseline pixel density of a line segment with height difference of 4.

Slope:

We explored the potential effect of slope type as well. When the left end of a line was lower than the right end, we categorized the slope as an *increase*, and *decrease* otherwise.

3.2.2 *Apparatus and Materials*

We utilized a smartwatch IMACWEAR M7, with 1.54" and 240×240 resolution display. The entire screen was used, so the participants were not distracted by the title and the notification bar. A Targus AKP03CA Bluetooth keypad was utilized to capture user input (Figure 3.6). Participants were instructed not to touch the display. The Bluetooth keyboard was used to enter responses to mitigate clutter on the watch.

3.2.3 *Compression Values*

The baseline graph is a graph with zero compression. Furthermore, to explore the lower approximate compression threshold, we investigated the *Compression Level* effect at 75%, 50%, and 25% for each *Compression Type* on error rates and response times. Note, when the graph is compressed on the x-axis to CL%, it indicates that the baseline graph was x-axis compressed by an amount of CL% on the indicated axis. This means, for instance, the bottom of the 25% x-axis compressed graph is shorter than the bottom of the 75% x-axis compressed graph. In Figure 3.2, each column (left to right) displays the baseline, xy-axis, y-axis, and x-axis compression.

Each row displays (top to bottom) 75%, 50%, and 25% compressed sizes, with height difference of 4.

3.2.4 *Study Design and Procedure*

We investigated three factors: *Compression Type* (xy-axis vs. y-axis vs. x-axis), *Compression Level* (75% vs. 50% vs. 25%) and *Slope* type (increase vs. decrease). The *Compression Type* was a between-subjects factor while the *Compression Level* and the *Slope* were within-subjects factors. Upon their arrival, participants were randomly assigned to one of three compression conditions (x-axis, y-axis, or xy-axis). After the instructions and signing the consent form, participants engaged in line graph reading tasks. Upon completion, participants received a $15 gift card. The entire session took approximately 60 minutes on average.

For each height difference (1, 2, 3, and 4) of each *Compression Level* (25%, 50%, and 75%) there were 15 graphs with positive slope and 15 graphs with negative slopes. For *Height* difference of 0, there were 30 graphs. Thus, each participant processed 150 graphs. There were four blocks for each session (100% or baseline, 75%, 50%, and 25%). The height difference (e.g., 2) and the slope direction (e.g., negative) were randomized within each session. A Latin square design was applied to counterbalance the order of 4 blocks.

3.2.5 *Participants*

We recruited 36 participants (Female = 12) mostly from a local university (M_{age}= 26.11). Since the session was designed to take approximately 60 mins, we chose a between-subject design to avoid potential cognitive fatigue.

3.2.6 *Collected Data*

Participants' response time (RT) and error rates (ER) were collected. Response was entered on the USB keyboard as quickly as possible. The RT was measured in millisecond once the graph was displayed on a smartwatch until they pressed the enter button on a keyboard. For the ER, the ratio of participants' inaccurate to accurate response was used.

3.2.7 *Results*

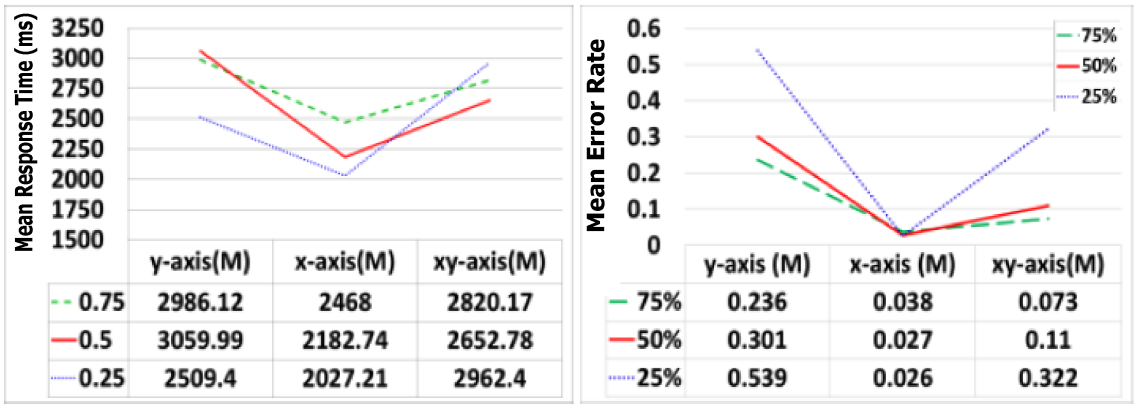
A series of mixed model ANOVAs were conducted throughout, unless otherwise specified. The between factor was the *Compression Type* (xy-axis vs. x-axis vs. y-axis), and within factors were *Compression Level* of the graph (75% vs. 50% vs. 25%), the *Height* difference between point A and B (1 vs. 2 vs. 3 vs. 4), and the Slope (increase vs. decrease). Mauchly's sphericity test was used to check the sphericity and whenever sphericity assumption was violated, Greenhouse-Geisser correction was applied. For the interpretation of effect size, Cohen's guideline was followed (0.01 = small, 0.06 = medium, 0.13 = large [21]. Small sample size can cause significant issues such as low statistical power, low producibility, and

non-normal data. Shapiro-Wilk test was used to check the normality of the data and although our data was not normally distributed, which means we can use data analysis methods designed for non-normally distributed data, we had a large enough sample size and thus, we did not expect any major issues [79].

Response Time (RT):

No significant main effect was found for *Compression Type*, *Slope* type, nor *Compression Level* ($p > .05$). As expected, a significant effect for *Height* difference was found however; $F_{1.8,49.4} = 102.03$, $p < .001$.

Further, a *Compression Type* $\times$ *Compression Level* interaction effect was found; $F_{4,54} = 2.73$, $p < .05$, $\eta_p^2 = .17$). Post hoc pairwise comparisons yielded that at 50%, on average, participants in the x-axis condition responded faster than those in the y-axis condition, with a mean difference of 500ms ($p < .05$). The same pattern was found at 25% level, with a mean difference of 1063ms, ($p < .05$). (See Figure 3.3, Left).



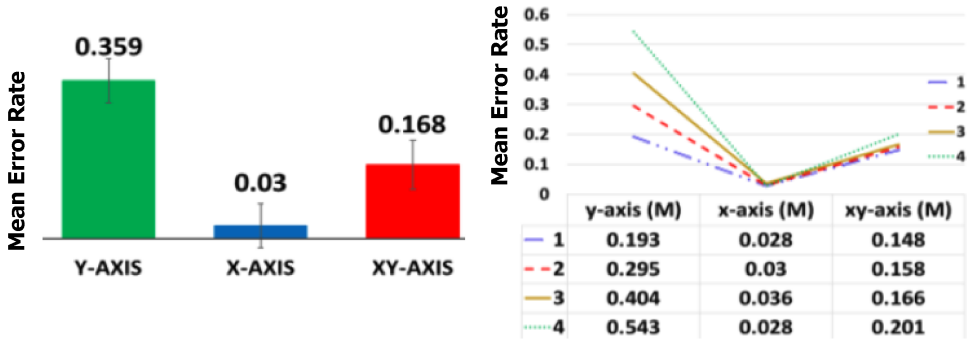
	y-axis(M)	x-axis(M)	xy-axis(M)
0.75	2986.12	2468	2820.17
0.5	3059.99	2182.74	2652.78
0.25	2509.4	2027.21	2962.4

	y-axis (M)	x-axis (M)	xy-axis(M)
75%	0.236	0.038	0.073
50%	0.301	0.027	0.11
25%	0.539	0.026	0.322

Figure 3.3: (Left): Study 1, Mean response time: Interaction between compression type and Compression Level. (Right): Study 1, Interaction effect on error rate; Between compression type and compression size.

Error Rate (ER):

We found significant effects for *Compression Type*; $F_{2,31} = 42.67$, $p < .001$, $\eta_p^2 = .73$, as well as for within factors, *Compression Level*, $F_{1.76,54.40} = 57.58$, $p < .001$, $\eta_p^2 = .65$), *Slope* type, $F_{1,31} = 14.21$, $p = .001$, $\eta_p^2 = .75$, and *Height* difference $F_{1.75,54.18} = 11.60$, $p < .001$, $\eta_p^2 = .27$. For the *Compression Type* effect, further pairwise comparisons were conducted. The x-axis compression yielded the lowest ER, followed by xy-axis and then the y-axis compression (Figure 3.4, Left). For the *Compression Level* effect, further post hoc analysis yielded that the ER did not differ between 75% and 50% ($p > .05$), while the ER at 25% was higher than that at 50% ($p < .001$), and at 75% ($p < .001$). Finally, for the *Height* difference effect, the largest difference (i.e., 4) was different from the rest ($p < .001$) while the rest did not differ ($p > .05$). Regarding the interaction effects, a significant *Compression Type* $\times$ *Size Interaction* effect was found on ER, $F_{3.67,56.84} = 16.42$, $p < .001$). A post-hoc analysis yielded no significant results for x-axis compressed lines ($p > .05$) while *Compression Level* effect on xy-axis and y-axis compression conditions were found ($p < .01$). This indicates that an x-axis compression is the most robust one against errors when time-series graphs are compressed. Their ER did not vary even when the line was compressed to 25% (See Figure 3.3, Right). Next, *Compression Type* $\times$ *Height difference* interaction effect was also found ($F_{6,93} = 7.04$, $p < .001$, $\eta_p^2 = .31$; Figure 3.4, Right). A simple effect analysis revealed significant *Height* difference effects ($p < .001$), but only for the y-axis compression. For x-axis and xy-axis compression style, height difference did not have significant effects ($p > .05$). Combined with the last finding (i.e.,robustness of x-axis compression), this finding confirms the stability of x-axis compression, in particular.



	y-axis (M)	x-axis (M)	xy-axis (M)
1	0.193	0.028	0.148
2	0.295	0.03	0.158
3	0.404	0.036	0.166
4	0.543	0.028	0.201

Figure 3.4: (Left): Study 1. Mean error rate by compression type. Error bars represent 95% confidence interval. (Right): Study 1. Interaction effect on error rate; Between height difference and compression type.

3.2.8 *Baseline vs. X-axis Compression*

Since x-axis Compression Type yielded the most favourable results for both RT and ER, we now compare the baseline (i.e., no compression) against an x-axis compression.

Response Time:

When the *Compression Level* on the x-axis was 75% or 50%, ANOVAs found no difference in Error Rate (ps $>$ .05). When the Compression Level was at 25%, however, a significant difference emerged with a large effect ($F_{1,21} = 4.23$, $p = .05$, η_p^2 .17). Participants responded significantly faster in X-axis Compression ($M = 2027.21$; $SD = 405.50$) than in baseline condition ($M = 2815.00$; $SD = 1259.99$). This indicates a potential benefit of X-axis Compression again. Generally, participants can respond faster towards x-axis compressed graphs than the baseline graphs when it is small.

Error Rate:

We also found significant effects for different height judgements for all *Compression Levels* ($ps < 0.05$) on Reaction Time. Again, all the significant results indicated the potential benefits of x-axis *Compression Type* ($rs > .75$).

3.2.9 *Study 1 summary*

For the RT, x-axis compression was better than other methods when the graph was compressed to a smaller size. Further, the results from the ER analyses pointed to more robust judgments with x-axis compression even at 25%. Compared against the baseline, the x-axis compression resulted in better performance across reaction time and error rate. Our assumption is that because of the nature of the task, comparing the height difference between two data points, compressing the line graph along the x-axis reduces the distance between two data points. This means it will be more accurate and faster to identify the difference between these two selected data compared to other compression techniques. Our outcome contrasts with the proposed practice of Sparklines, which compress along the y- dimension to fit in word size chunks. We use this largest density (25%) compressed along the x-axis for G-Sparks.

3.3 STUDY 2: G-SPARK

Since the x-axis compression consistently yielded the most favourable outcome in Study 1, we propose G-Sparks, a condensed line graph to represent the densest compression. At 25%, each pixel in G-Sparks represented one sample point from

the heart rate data set we used (Figure 3.8). The goal of this study was to assess this aggressive *Compression Type*. This is the most aggressive level of compression possible, for representing every data sample. As indicated earlier, three of the most common perceptual tasks with line graphs are: (1) max/min detection; (2) slope value detection; and, (3) the value differences between two or more data points (Figure 3.5). We can place these tasks along a 'glanceability' continuum according to the degree of task difficulty, with peak estimations being highly glanceable, and size and slope judgments less glanceable. We use these tasks to assess G-Sparks.

Figure 3.5: Tasks falling on a 'glanceability' continuum, with tasks such as min/max detection being highly possible by glancing, while slope degree and size estimations being less 'glanceable', i.e. needing cycles to compute the difference. We use these tasks to assess whether the x-axis compression style can still enable proper judgement.

3.3.1 *Apparatus and Materials*

We used the same device as in Study 1 (Figure 3.6). We used actual heart rate data where each sample point represented one second, and all graphs contained at least 2000 data points. With our apparatus, and with the Baseline and G-Sparks density described earlier, we could fit 50 and 200 data points, respectively, on the smartwatch in any instance. To see the remaining points users could flick left or right.

Figure 3.6: The smartwatch and Bluetooth keypad used in both studies. The arm with the watch was placed on the table.

Heart rate data has different patterns related to various activities, such as resting and sleeping, with more stable heart rate, compared to other activities such as workout with more changes in the data. Users usually are looking for parts of the graph with fluctuations and changes in the data. Therefore, for this study, we were more interested in heart rate data with more fluctuations (e.g., doing workout). Also, our standard tasks in this study are designed in a way that there should be fluctuations in the data.

3.3.2 *Procedure*

Upon their arrival, participants were randomly assigned to G-Sparks or to the Baseline condition. After reading instructions and signing a consent form, participants engaged in the assigned graph reading tasks. Upon completion, participants

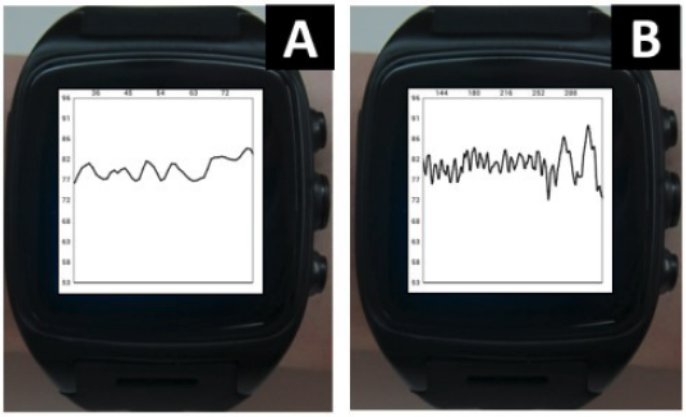


Figure 3.7: (A). Baseline line graph, and (B). G-Sparks.

received a $15 gift card. On average the session lasted 60 minutes.

Max/Min Detection:

All participants saw 20 line graphs, and they flicked to navigate the entire graph. Participants were asked to tap the highest peak value on the smartwatch display. Once they identified the peak, they pressed enter on the keyboard to move to the next line graph. They were allowed to flick right and/or left as many times as they needed and were able to change their answers until they moved to the next graph. The same method was used for minimum trough detection (i.e., to identify the lowest data point). All the graphs were unique.

Slope Estimation:

Participants saw two marked slopes in a line graph, and they were asked to select the steeper one. These slope stimuli were selected based on the following rules; (i) the two slopes were not in the same frame, i.e., users had to flick the viewport at least once, (ii) one slope was always steeper than the other. Participants repeated

this task 10 times.

Height Difference Detection:

Participants saw 2 red dots on a graph, and they assessed the value difference on the y-axis. Analogous to Study 1, we implemented this task in a way that the value difference between two data points was between 0 and 4 units. For each value difference, there were 10 tasks resulting in 50 trials for the height difference task. These paired points were selected based on the 2 conditions: (i) the height difference of these points should not exceed 4 units, and (ii) the two points were not seen on the same frame.

Flicking Frequency:

With at least 2000 data points in our samples, the chart was large enough to require that all the participants flick through to process the entire graph. We also recorded participants' flicking frequency.

3.3.3 *Study Design*

In this experiment we used a between-subject study design. Participants in the experimental condition were exposed to a series of G-Sparks, while participants in the control group were exposed to a series of baseline graphs (i.e., no compression, Figure 3.7). In each condition, we used different tasks to evaluate and compare G-Spark with the control group. These tasks include max/min detection, slope estimation, and height difference detection. A balanced Latin square of size four was used to prevent the order effect of tasks on the result. In the baseline condition,

the number of data points presented within a frame was 50. All the data points are presented equidistantly and the distance between each two consecutive data points is 4 pixels on the x-axis. For the G-Spark, due to the 25% compression of the baseline, the interval between the data points was 1 pixel on the x-axis. Thus, we had 50 data points for the baseline in each screen (Figure 3.8). Note that graphs in both the baseline and G-Spark conditions span the entire width of the smartwatch display in the experiment. Therefore, we expect to see improvements in the G-Spark graphs over the baseline condition for some of the tasks. For instance, as the distance of each two data points is reduced due to the compression along the x-axis, we will most likely have fewer flicks in general. Also, for tasks such as the data value comparison task, we predict some improvements in accuracy, response time and the number of flicks due to the reduced distance of selected data points.

In the beginning of each session, each participant was randomly assigned to one of the two conditions (12 participants per condition). The order of tasks for each participant was based on our balanced Latin square. After performing each task, participants had the option to have a break for 2 minutes. After the instructions were given, participants had a trial session. The tasks were the same across both conditions.

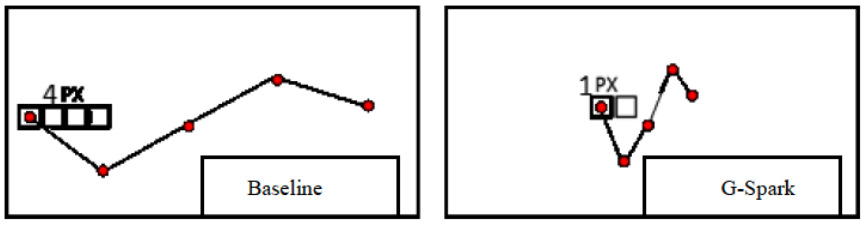


Figure 3.8: Representing the x-interval of data points in baseline line graph and G-Spark

3.3.4 *Participants*

To avoid bias we recruited 24 new participants (Female = 10) mostly from a local university ($M_{age} = 25.75$).

3.3.5 *Collected Data*

Same as in Study 1, we collected participants' response time (RT), as well as the error rate (ER). For the flicking frequency, participants' flicking response was recorded on the smartwatch. Participants could go back and forth as frequently as they needed.

3.3.6 *Results*

After normality was ensured by using Shapiro-Wilk test, independent sample t-tests were conducted throughout, except for the Height Difference Identification. For the Height Difference Identification, a mixed between-within subjects ANOVAs were conducted, with condition (baseline vs. G-Spark) as between factor and the height difference as a within factor (0 vs. 1 vs. 2 vs. 3 vs.4).

Minimum Point Detection

Response time (RT):

Consistent with the results of Study 1, the mean RT for the participants in the G-Sparks condition was significantly shorter ($M = 18{,}407$ms, $SD = 6197$) than the

mean RT for the baseline counterpart (M = 27, 432 ms, SD = 13343; t_{22} = -2.13, $p < .05$). The effect size was large ($\eta^2 = .17$), indicating the magnitude of the differences in the means (9025ms) was large [12].

Flicking:

Participants in the G-Sparks condition flicked fewer times (M = 19 flicks, SD = 6) compared to the participants in the baseline condition (M = 44 flicks, SD = 18; $t_{13.38}$ = -4.55, $p < .01$). The effect size was large (η^2 =.49) indicating the very large magnitude of the differences in the means (mean difference = -25.08, 95% CI [-36.96, -13.20]).

Error rate (ER):

Independent sample t-tests indicated significantly higher error rate in the baseline condition (M = .25, SD = .16) than with G-Sparks (M = .06, SD = .08; t_{22} = -3.74, $p < .01$ - two-tailed). The magnitude of the differences in the means (.20) was again, very large η^2 =.40. For minimum point detection, therefore, participants in the G-Spark condition flicked fewer times and completed the tasks faster with fewer errors than the participants in the baseline condition.

Maximum Point Detection

Response time (RT):

Similar to minimum detection, the mean RT for participants with G-Sparks for maximum detection, was shorter (M = 18533ms, SD = 6005.91) than the baseline (M = 28636ms, SD = 8960; t_{22} = -3.16, $p < .01$; two-tailed). The magnitude of the

differences in the means (9829ms) was large, η^2 =.31.

Flicking:

Again, analogous to the minimum detection, participants in the G-Sparks condition flicked fewer times (M = 20 flicks, SD = 7.24) compared to participants in the baseline (M = 46 flicks, SD = 14.24; $t_{16.33}$ = -5.66, p < .001; two-tailed). The magnitude of the differences in the means (26) was very also large η^2 =.59.

Error rate (ER):

We observed a higher error rate in the baseline condition (M = .28, SD = .10) than with G-Sparks condition (M =.06, SD = .06; t_{22} = -.6.3, p < .001; two-tailed). Further, the magnitude of the differences in the means (.21) was very large η^2 = .65. Once again, participants in the G-Sparks condition flicked fewer times and completed the tasks faster with fewer errors than the participants in the baseline.

Slope Estimation

Response time (RT):

Consistent with the results of minimum and maximum points detection, the mean RT for the participants in the G-Sparks condition was shorter (M = 17212ms, SD = 6049) than the mean RT for the baseline counterparts (M = 26998ms, SD = 6582; t_{22} = -3.79, p < .01; two-tailed). The observed mean difference was large (η^2 =.40) (9786ms).

Flicking:

Participants in the G-Sparks condition flicked fewer times (M = 13 flicks, SD =

4) compared to the participants in the baseline condition (M = 32 flicks, SD = 5; t_{22} = -9.19, p < .001; two-tailed). The magnitude of the differences in the means (19 flicks) was very large η^2 = =.79.

Error rate (ER):

For the steeper slope detection as well, the ER was higher for the baseline condition (M = .32, SD = .14) than for G-Sparks (M = .07, SD = .05; t_{22} = -5.82, p < .001 - two-tailed). Further, the magnitude of the differences in the means (.25) was large η^2 = .61.

Height difference Identification

Response time (RT):

There was a significant interaction effect between height difference and condition, $F_{4,19}$ = 9.17, p < .001, with a large effect; η^2 = = .66. We thus investigated the simple effects of height difference for each condition. For G-Sparks, participants' RT was the longest when there was no height difference (M = 22381ms, SD = 4857) or when there was four (i.e., maximum) unit differences (M = 19556ms, SD = 5793, ps < .05). Participants' RT were equally shorter when the unit differences were one, two, or three. For the baseline condition as well, participants' RTs were the longest when there was no difference (M = 35101ms, SD = 5704), followed by the 4 unit, maximum difference (M = 29243ms, SD = 6243). Again, participants' RT was shorter when the unit differences were 1, 2, or 3. Further and importantly, participants' mean RT was consistently shorter in the G-Spark condition (ps < .005).

Flicking:

A mixed between-within subjects analysis of variance was conducted to assess the impact of different types of graphs (baseline vs. G-Spark) on participants' flicking frequency, across five different height difference levels (0,1,2,3,4). First, we found an interaction effect between the presentation condition and the height difference, $F_{4,19} = 3.10$, $p < .05$, with a large effect, $\eta^2 = 40$. Next, we investigated the simple effects of height difference level for each condition. There was no height difference effect in the G-Sparks condition, indicating the flicking frequency did not vary based on the height differences with the G-Sparks condition ($ps > .05$). For the baseline condition, however, participants' mean flicking frequencies was the highest when there was no height difference ($M = 39$ flicks, $SD = 6.90$) or when there was 4 unit/maximum difference ($M = 36.20$, $SD = 6.89$). Participants' flicking frequencies were the lowest when the unit differences were 1, 2, or 3. Further and understandably, for each height difference, G-Sparks constantly exhibited a smaller flicking frequency ($M = 13$ flicks, $SD = 4.46$) than the baseline ($M = 33$ flicks, $SD = 5.44$; $F_{1,22} = 164.87$, $p < .001$), with a very large effect $\eta^2 = .88$.

Flicking:

We investigated the main effect of condition. Participants' mean ER was higher in the baseline condition ($M = .52$, $SD = 20$) compared to the G-Sparks condition ($M = .52$, $SD = .17$; $F_{1,22} = 19.35$, $p < .001$), with a very large effect $\eta^2 = .47$. For both conditions, when the height difference did not exist (i.e., 0) or one, ER was smaller ($ps < .05$), but as the height difference increased, the ER also increased. There was no significant interaction effect ($p > .05$).

3.4 DISCUSSION

As expected, relative to zero-compressed graphs, G-Sparks led to better overall outcomes. Compared to the baseline graphs, G-Sparks yielded shorter response times, with fewer flick operations. Further, overall error rate was lower with G-Sparks compared to the baseline. Altogether, these results indicate that G-Spark, a line graph to present each sample point in one pixel, has potential for displaying line graphs on small displays.

We used the most fundamental tasks related to reading and understanding line graphs, representing heart rate data, as one of the common time-series data collected by smartwatches sensors. Since all time-series data are similar to each other and can be represented by line graphs, the result of this study can be generalized to other time-series data. For instance, a similar approach can be used for representing burned calorie, walking speed, galvanic skin response, and body temperature of the user, for a specific period.

3.4.1 *Result Summary*

We used a highly aggressive compression style, where each pixel represents one sample point in the data. We referred to this line graph as G-Sparks. Our results point at condensing line graphs along the x-axis as these lead to fewer errors and minimal interactivity. Altogether, our results indicate that x-axis compressed line graphs offer robust graphical judgements on smartwatch displays. It is worth mentioning that fewer flicks can be key for smartwatch experiences [103].

3.4.2 *Relation to Sparklines*

We drew inspiration from Sparklines [108] a method for compacting a line graph within words in a document. Interestingly, Sparklines compresses graphs on the y-axis to make the graph fit as a word in a text passage. This may indeed not severely affect global trend understanding. However, for specific size judgments, we find that x-axis and even xy-axis compressions are best suited for visual exploration tasks. Such a compression creates space to include additional details on the watch display. Although Sparklines are useful in various applications, we demonstrated that an x-axis compression, or G-Sparks, is better suited for fundamental tasks related to reading and understanding line graphs on smartwatches.

3.4.3 *Relation to Sparklines*

We offer the following design recommendations:

- Compressing line graphs along the x-axis will facilitate performing visual tasks and reduce flicking on smartwatches;
- G-Sparks is well applicable to data collected from fixed sample rates for efficient graph interaction.

3.4.4 *Applications*

Various applications can benefit from designs such as G-Sparks. For example, G-Sparks can be combined with other data visualizations to convey additional

insights to the users. This can provide very dense and compact representations for data such as heart rate, sleep quality, and breathing patterns all together on the small smartwatch display (Figure 3.9, a). In more complex scenarios, G-Sparks can also represent the elevation of a selected route in a map, from the beginning point to the ending point (Figure 3.9, c). As such, a jogger may make decisions about their speed according to the visible elevation. Furthermore, temperature and precipitation patterns can also be added to help users make suitable decisions. It's also possible to use G-Sparks in round watch faces (figure 3.9, b) with more complex line graphs such as horizon graphs and stack graphs. While we evaluated our approach on a rectangular display, we believe they also apply to circular screens (see above). In such cases, the line graphs can be confined to 'tight' positions, such as at the bottom or top of the screen. Furthermore, our results can be combined with other approaches, such as Horizon graphs to further condense our representations along the y-axis in case of including additional data.

Figure 3.9: G-Spark applications. a) Adding more details and information, b) using G-Sparks for more complex line graphs in round faced smartwatches, and c) can be embedded in different applications, helping users with decision-making tasks.

3.4.5 *Limitations*

One limitation concerns our focus on values read from the y-axis but not the x-axis, which represents the temporal aspect in line graphs. To explore the generalizability of our results, future work is needed. Future work is also necessary to look at other perceptual tasks. Correspondingly, the generalizability of the results on different smartwatch displays, such as those that are circular, is unknown. Furthermore, when condensing a line graph, interactivity, such as selecting a specific point is challenging. Future work will also investigate suitable interactions for G-Sparks.

3.5 CONCLUSION

We investigated methods to condense line graphs on smartwatches, while maintaining the visibility during relatively swift graph reading activity. Inspired by Sparklines [48], our first study focused on comparing the benefits of three different *Compression Types* (i.e., x-axis, y-axis, and xy-axes compression) with simple line segments. We repeatedly found that compressing a line graph on the x-axis yields more accurate and faster response, compared to the y-axis and xy-axis compression. Moreover, even when the x-axis compression was compared against the baseline, x-axis compression generated more favourable outcomes. That is, somewhat unexpectedly, compressed graphs triggered more accurate and faster responses than uncompressed graphs. We finally introduce G-Sparks, a dense compression of line graphs of up to one pixel per sample point along the x-axis. In a second study we find that compressing line graphs on the x-axis yields better performances (i.e., fewer errors and faster response time) when users performed

estimation tasks, when compared to a non-compressed line graph. We offer design recommendations to smartwatch application designers and propose ways to integrate such compressed graphs in smartwatch applications.

4 SF-LG: SPACE-FILLING LINE GRAPHS FOR VISUALIZING INTERRELATED TIME-SERIES DATA ON SMARTWATCHES

4.1 EXPLORATORY REVIEW OF SPACE UTILIZATION ON SMARTWATCH APPS

We first examine how current smartwatch applications utilize space when presenting data visualizations. Our analysis included highly ranked fitness and health-related smartwatch apps from Google Play and the App Store (12 watchOS, 2 Android Wear, and 8 Android Wear & watchOS apps in total). To do this, one of the authors collected the top 40 ranked apps in the fitness tracking category, which include capturing sleep quality, heart rate, workout duration, and water consumption, from both Google Play and the Apple App Store (top 20 ranking apps on each platform). These apps were specifically designed for smartwatches. We installed and tried all 40 apps on two smartwatches, an Apple Watch series 4 with a 44mm display, and an Android Macwear M7. From these 40 apps, only 22 of them used data visualization techniques to represent the data and the rest only used text or icon to represent fitness information. This result confirms the findings reported by Islam et al. [49] that using charts is not currently a common way to represent data on smartwatch applications and needs more investigation. We then extracted all visualization techniques from 22 selected apps by using the built-in screen capture feature on both smartwatches, and measured how much of

the entire space was occupied by the visualization techniques. We also measured the empty space used throughout these screens. Using Inkscape [1] we measured the exact proportion of the occupied space over the empty space by calculating the rectangular space an element occupies on the screen (e.g., Fig 4.1). The empty space is equal to the area of the entire screen minus the area occupied by all elements.

Our review focused solely on apps available for rectilinear smartwatches as this was our target platform in this initial study. The choice of a rectilinear watch was to even out our comparison between Apple and Android apps. In total, we reviewed 38 interfaces from 22 smartwatch apps (some apps had more than one screen). We then explored each app by examining (i) space usage and (ii) varieties of chart types used.

4.1.1 *Space utilization*

Based on the selected apps, we explored space utilization on rectilinear smartwatch displays to understand how designers organize visual data. In particular, we aimed to understand the interplay between the space allotted to the primary visualizations and auxiliary information, such as labels, icons, or other charts. To this end, we first computed the entire area of the interface and the area allotted to visual elements on screen. We further categorized on-screen content into three common visual elements we observed on such interfaces: data visualizations/charts, text, and icons. The space allotted to each element, the free space without visual elements, and the occupied ratio (percent of the interface area with visual elements on it)

1 https://inkscape.org/

were manually computed based on screenshots we captured. Figure 4.1 shows these measures on four selected smartwatch visualization interfaces.

We observed a high degree of variability in the amount of free space for the 38 interfaces (Figure 4.2.left). In total, more than half (55%) of the interfaces we examined had between 40% to 70% of free space. Further, no app had free space that was larger than 70%. We note that free space is important for improving the legibility of content but we were equally surprised to see such a large amount devoted to no visual content.

There was also a large variability in space occupied by charts (Figure 4.2.right). In the majority of the apps (60%) charts occupied less than 50% of the available space, whereas only 26% of the apps used more than 70% of the available space. 89% of the interfaces used icons that occupy between 0-10% of the interface space, whereas only 10% of the interfaces had icons that occupied between 10–20% of the space. Interestingly, the analysis of text showed text elements occupied between 0% to 30% of the total space across all interfaces. About half (47%) of the apps used text that occupied 10–20% of the screen. While specific to the apps we surveyed, these results provide insightful context on how designers distribute balance visual content and free space on the small smartwatch screen.

4.1.2 *Chart types*

The reviewed apps most commonly employ Bar charts, used in 50% of the apps, followed by Donut (26%), Line (24%), and Scatter (5%) charts. We also looked at the number of charts used in each interface. The analysis showed that 95% of

2 Runtastic Run, Mileage Tracker from Apple App Store

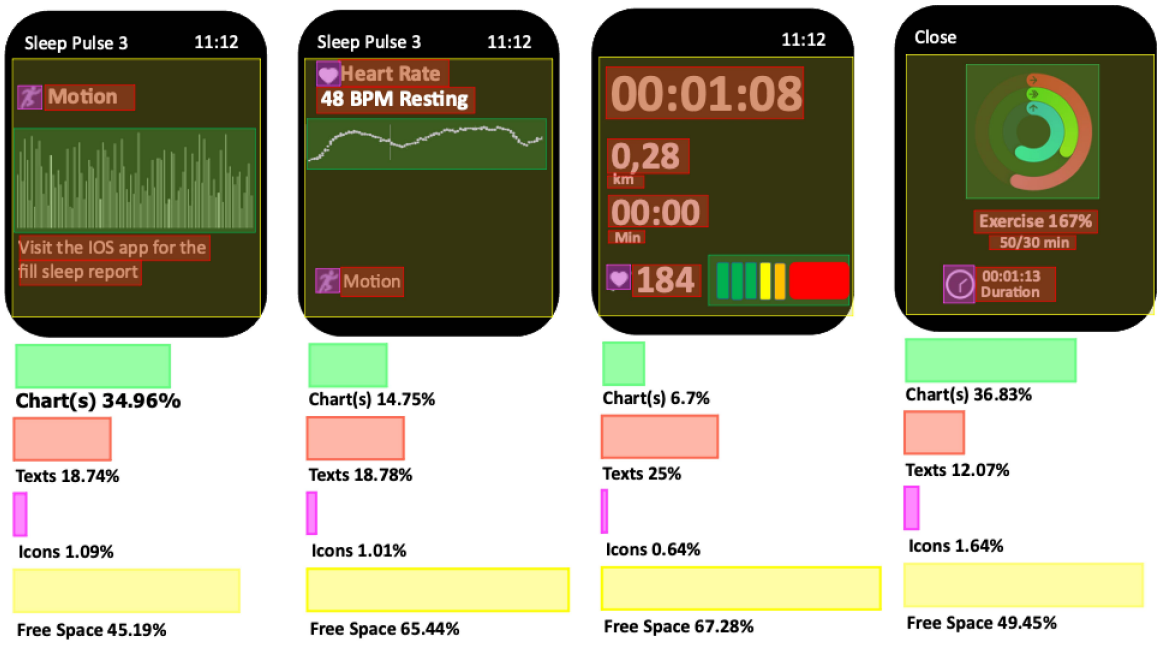


Figure 4.1: Space usage analysis on four illustrative visualization interfaces across two applications[2], indicating the percentage of free space and of space occupied by each visual element: charts, texts and icons .

the visual interfaces used one graph compared to only 5% that used up to four charts. This also confirms the results reported by Islam et al. [49] that the average number of chart representations per watch face was nearly one across a wide range of smartwatch applications. Our result shows that most existing smartwatch applications deploy separate visual displays to represent charts which means each visual display represents a single chart. For instance, a line chart can be used to describe heart rate data, and in the following display, a radial bar graph to show the proportion of different activities the user did in a day.

Figure 4.2 shows that text can also represent data to the smartwatch users, mainly in the form of a number, besides the main graph. However, text can only convey one piece of information. For instance, a text can represent the user's current heart rate, walking pace, or body temperature. However, more information

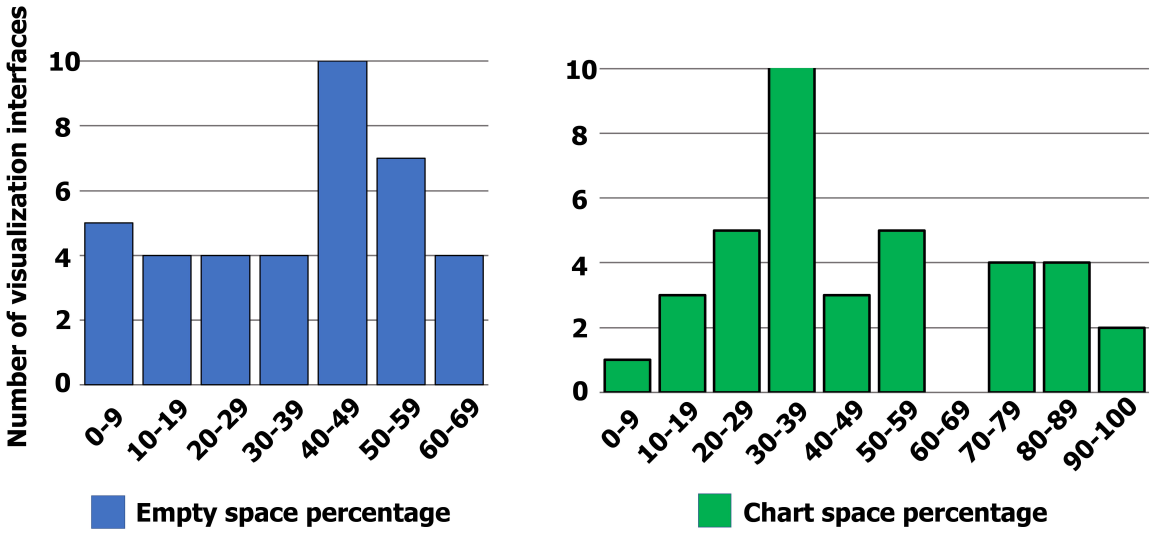


Figure 4.2: (left) Distribution of free space and (right) Distribution of space occupied by charts in the reviewed apps (N = 38).

can be transferred by using visualization techniques. A line graph represents the data related to the user's current condition, but also the trend and information about the history of the data. In general, reading and understanding a graph can take more time than reading a single number. This means that text could be helpful for queries related to the user's current condition, which does not require any information about the history of the data. As a number can be read faster than a graph in general, both text and chart can be used together. Using a smaller text size and embedding the text within a graph can be helpful to take advantage of the limited space on a small screen of smartwatches to represent both text and graphs.

Based on our analysis, we summarize the following key observations (OB), which are closely tied to the review of apps we examined:

- **OB1:** The majority of the apps use little space for their key charts, and do not take advantage of the entire display.

- **OB2:** Most interfaces have a considerable amount of free space. This potentially implies that the overall space utilization could be further optimized to include auxiliary content to the primary data chart.

- **OB3:** Most apps that include more than one visualization from multiple sources (e.g., a radial graph for steps walked and a bar graph of burnt calories) require users to flick between these visual displays. Interlinked charts are rarely presented on a single screen.

- **OB4:** Most apps reviewed used Donut charts rather than Line charts to represent time-series data. However, earlier work [89] has shown Bar and Line charts are best suited to visualize trends and numerical values over time.

- **OB5:** We found that many of the icons augment textual descriptions (see Figure 2; heart rate icon and label). These are important in describing the charts and could be further optimized (but we leave this for future work).

- **OB6:** A text or a number can be embedded in graphs to convey one piece of information related to the current condition of the smartwatch user. Using smaller text can help the designers embed text within graphs on such small displays without minimizing the size of the chart.

These observations suggest that chart interfaces can be further optimized to use the available space. Either by (i) enlarging the chart or (ii) augmenting the chart with either additional information/graphs. Furthermore, we believe additional space can be made available if the area devoted to the chart is optimized. For this reason, we next seek to expand the available space by (a) simplifying the chart and then (b) augmenting the freed space.

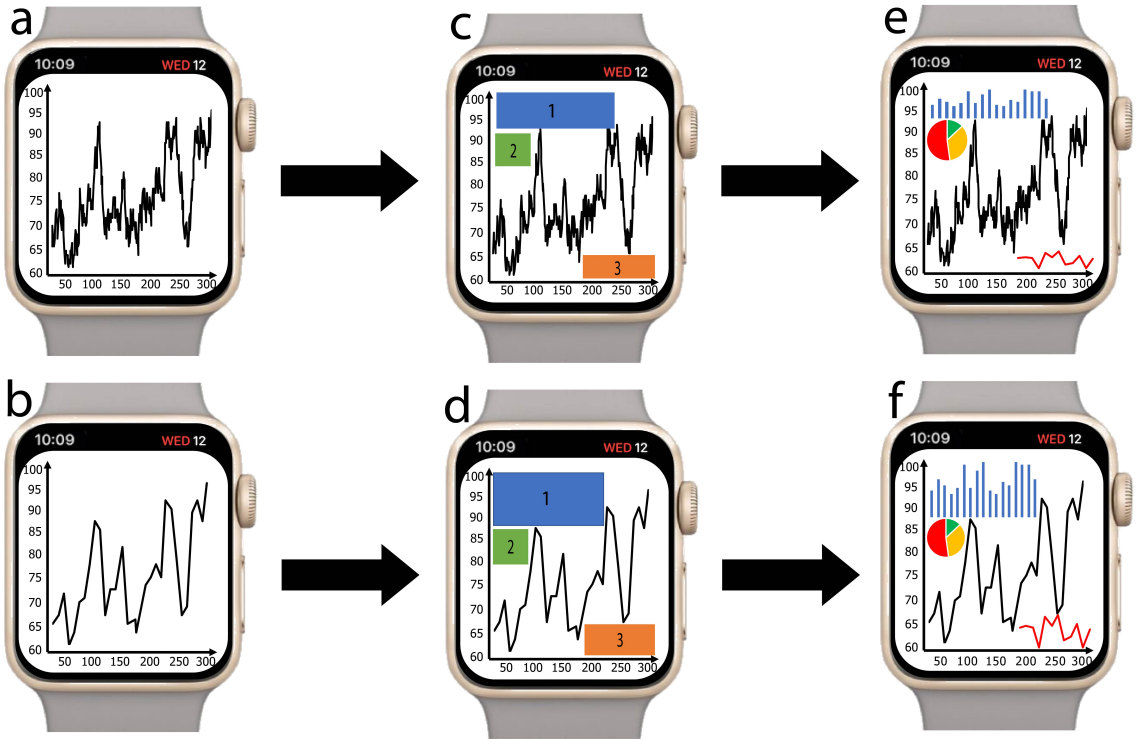


Figure 4.3: (a): A line graph representing heart rate over six minutes and including 300 data points; (b): we simplify the line graph in (a) using the Space-Filling Line Graph (SF-LG) technique; (c) & (d): the available space created around the line graph can be used to fill auxiliary, interlinked information. In our approach, we can provide the algorithm a priority to position the information. The labels "1", "2", "3" represent the priority assigned to those components which are given the maximal space; (e) & (f): embedding auxiliary and interrelated graphs that augment the primary line graph becomes possible based on the reorganization of the available space

4.2 SPACE FILLING LINE GRAPH (SF-LG)

We introduce a Space-Filling Line Graph technique that has two elements, (i) a graph simplification algorithm and (ii) efficient embedding of auxiliary data. We first present our simplification approach which frees screen space and then describe how we embed supplemental charts compared with non-simplified graphs (Figure 4.3). Similar to PIP, we also focus on data points that have "significant meanings",

such as peaks, maximum/minimum(s), and outliers. In SF-LG, we look for data points with the highest average distance to all other data points. This provides us with a subset of data points in the line graph that have significant meaning (i.e., anomalies).

We calculate and store the average of the Euclidean distance of each point p_i such that ($i \in [0, N]$, where N is the sample size) for the entire set of points (P) in the graph. A point with higher average distance indicates that this data point is relatively far away from the rest of the dataset. This causes a challenge because the right-most (and left-most) points in the graph have no neighbours to its right (or left). Furthermore, a large difference in x-values of the right- and left-most data points in the graph makes the average distance of these points artificially larger. This implies that data points with very high and very low x-values will have a higher average distance compared to data points at the graph's center (i.e., data points with neither very high nor very low x-values). To resolve this, akin to the PAA approach, we define windows. With the windowing technique, we can select points with the highest average distance to all other data points, compared to all of its neighbours, for that same window (Figure 4.4). As such, we can calculate and select the essential points in each graph window. Algorithm 1 shows the SF-LG pseudocode and Figure 4.4 shows how SF-LG works.

Algorithmus 1 : Simplification algorithm.

Input : A set P, of points in the graph with x and y coordinates and window size

Output : A subset of P, representing data points with the highest average distance to all other data points

```
1  for each data point p_i in P do
2      Sum = 0;
3      for each data point p_j in P (i != j) do
4          EuclideanDistance = calculate Euclidean distance from p_i to p_j;
5          Sum = Sum + EuclideanDistance;
6      AVGDistanceArray[i] = (Sum / number of data points - 1);
7  Divide the set of data points into subsets (windows);
8  for eachwindow do
9      mark the point with the highest AVGDistanceArray[i];
10 return marked points
```

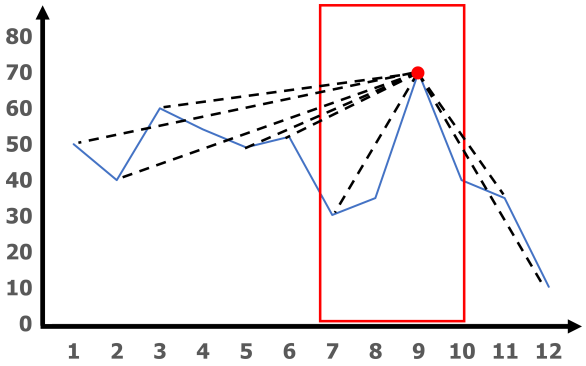


Figure 4.4: SF-LG calculating the average distance of the red data point in the red rectangle to all other data points in the graph leads to the highest average distance compared to the remaining two data points in the same window. This process is performed for all data points in all windows to choose the data point with the highest average distance.

4.2.1 *Comparing simplification techniques*

The aforementioned Sampling technique selects data points with fixed and specific intervals and does not consider the importance of data points. SF-LG solves this by focusing on data points that are more important in the dataset (e.g., anomalies and extreme points), similar to the PIP method. Furthermore, unlike Sampling, as we are using the windowing technique with actual data points in SF-LG, many visual features of the line graphs such as trends, patterns, extreme points, and the general shape of the graph are preserved.

Similar to PAA, increasing the window size (i.e., fewer windows) will reduce data points in the graph, which can affect the representation accuracy of the original graph. Because PAA uses the average of each window and not real data points, the PAA-simplified version of the graph could significantly differ from its original. However, preserving the original extreme data is often crucial, especially for biometrics (e.g., extremely high heart rate). In comparison, because SF-LG uses actual data points, there will be no distortion in the simplified graph.

SF-LG deploys the same concept used in PIP but with improvements aimed at representing and simplifying line graphs on smartwatches. In different windows, data points are selected that have close values (data points that are further away from the rest of the data) which can prevent extreme fluctuations in the graph. This has the effect of creating additional space (Figure 4.5) which can be used to present augmented information such as pictures and graph details (Figure 4.3.f). Furthermore, smartwatch hardware is very limited and for this reason we chose to compare PIP with SF-LG. Limited processing capabilities of smartwatches prevent application designers to deploy complex algorithms to visualize or analyze

the data. Using non-intensive algorithms such as PIP, and our SF-LG method, smartwatches can take advantage of these simplification techniques. To confirm this, we compared the processing time it takes for both SF-LG and PIP algorithms to simplify 30 data sets with 300 data points. To ensure everything is the same for both conditions, we used the same data sets for both algorithms. On average, it took 12.76 ms (min=11 ms, and max=22) for SF-LG and 17.25 ms for PIP (min=15 ms, and max=22) to simplify these graphs. This result illustrates how fast these algorithms are on smartwatches with limited processing capabilities to simplify large data.

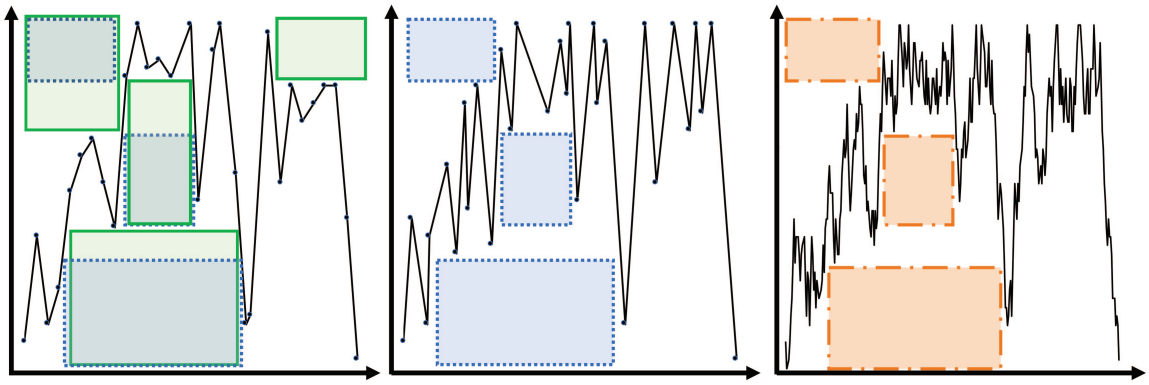


Figure 4.5: Visually comparing simplification techniques for line graphs: left) SF-LG, middle) PIP method, right) non-simplified graph. Green spaces in the left image represent the spaces created by the SF-LG method, overlapping with blue spaces generated by the PIP technique.

4.3 STUDY 1: SF-LG GRAPH SIMPLIFICATION

In Study 1 we compared the effect of simplification with SF-LG against two baselines, the Non-Simplified and PIP techniques. We chose PIP as one of our baselines for the following reasons: 1) it is one of the simplification methods that does not require computational power which makes it suitable to be implemented

on smartwatches with limited computational capability; 2) it has been commonly used to simplify health data, which is one of the main reasons smartwatches are being used, 3) it preserves many of the important data points of the original graph (e.g., maximum and minimum), and 4) considering the visual aspects of the graph to simplify the graph.

4.3.1 *Conditions*

Our evaluation included three visual conditions: SF-LG, Non-Simplified, and PIP. PIP was carefully selected for three important reasons: (i) PIP is one of the most widely applied simplification techniques; (ii) PIP is the only major technique that does not compromise crucial data points such as maximums and minimums; (iii) PIP is a simple algorithm which can be deployed by smartwatch applications due to limited processing and memory capabilities of smartwatches. Thus, having these conditions will allow us to investigate a) whether SF-LG can improve users' performance (i.e. accuracy and response time) compared with the Non-Simplified graph; b) whether SF-LG achieves at least equal performance compared with PIP, a popular simplification method. Other simplification techniques are also available but given the popularity of the PIP approach, and its ability to preserve salient data points, we use this in our exploration.

4.3.2 *Participants and apparatus*

Thirty (30) participants, 10 for each condition, ($F = 13$; $M = 17$; $Mage = 28.53$; 27 right-handed), with normal to corrected vision, were recruited via poster

advertisement at a local university. Participants completed the Ishihara colour blindness test [3] which confirmed that none were colour blind.

We used an IMACWEAR M7 smartwatch, with a 1.54" and 240×240 resolution display. To avoid any distraction by the title or notification bars on the screen, we used the entire screen for the study. For user input, based on [12], we used a Targus AKP03CA Bluetooth keypad (Figure 4.6) so we could solely focus on the visualization aspects without introducing possible confounds (e.g., fat-finger concerns) due to small displays. Participants were asked not to touch the display.

We chose not to control the orientation and the distance of the smartwatch, unlike [12] for three main reasons: 1) to preserve the authenticity of users' daily smartwatch usage without any artificial restrictions; 2) even if the distance was fixed at the beginning, the participants could still move their heads; 3) the discomfort that fixed distance could induce may impact the participants' performance.

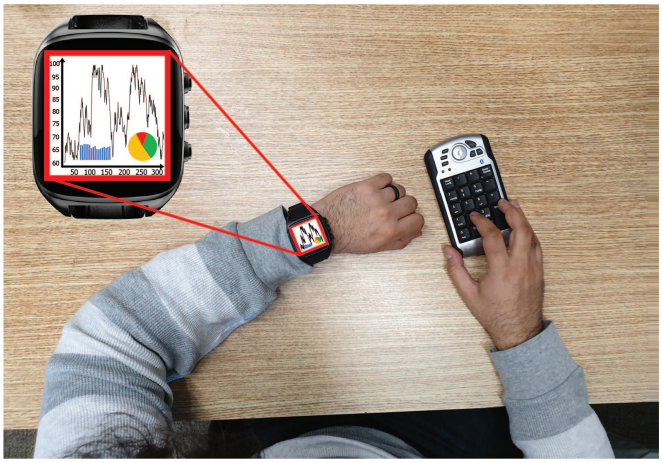


Figure 4.6: Apparatus and setup used in both studies.

3 https://enchroma.com/pages/color-blind-test

4.3.3 *Tasks*

There were five tasks in total, three of which were adapted based on previous studies (see Perceptual Tasks on Line Graph section 2.4). All tasks are described below. For each task, each participant was asked to process ten unique line graphs. Participants had to finish one task to be able to carry on to the next task. Each graph contained 300 data points. We used a balanced Latin square design for the study. Furthermore, to boost the generalizability of our results, we used real heart rate data as stimuli instead of synthetic data [30, 74].

Maximum/Minimum Detection:

In this task, participants were asked to detect the highest/lowest data points on each graph. Participants were asked to report the x-value of the max/min data point using the keypad, and pressed Enter to move to the next line graph. To avoid any confusion, each graph had only one maximum and one minimum data point.

Value Detection:

In each line graph, one data point was marked with a small red circle, and participants were asked to type the exact y-value of this point. This is a commonly performed task on line graphs (e.g., reading the exact heart rate, the number of steps, or the body temperature).

Value Comparison:

Two randomly selected data points were marked with small red circles in each graph. The participants entered the difference between them as measured on the

y-axis.

Trend Detection:

Participants were asked to indicate the overall trend of each line graph (i.e., an increase or decrease) by pressing the UP or DOWN arrow buttons on the keypad. We ensured that each graph contained an unambiguous trend. An ambiguous trend line is a trend line that is close to the flat (horizontal) line. This means that it is not clear if the trend is increasing or decreasing.

4.3.4 *Study design and procedure*

The study followed a between-subject design with one factor, Interaction Technique (SF-LG vs. Non-Simplified vs. PIP). After reading and signing the consent form, participants were randomly assigned to one of the three conditions. A between-subject design was selected to avoid potential cognitive fatigue because the entire session was designed to take longer than 30 minutes. For each condition (e.g., SF-LG, PIP and Non-Simplified), we counterbalanced the order of five tasks by using a Latin square design. Before each task, a research assistant explained the process and gave detailed instructions. Participants were asked to perform each task as quickly and accurately as possible. First, each participant was allowed practice trials with sample line graphs until they felt confident about performing each task. Participants were informed that all the data points' x- and y-values were integers (i.e., whole numbers), and not fractions (e.g., 56.5). Participants received a $10 gift card upon completion.

To collect post-study user preference, we presented participants with three graphs (SF-LG, Non-Simplified, PIP) made from the same datasets. Participants were asked to rank the graphs based on their preference. Also, participants' Accuracy and Response Time were recorded as dependent variables for further analysis.

4.3.5 *Results*

Shapiro-Wilk tests yielded that only the collected response time for the minimum detection task was normally distributed. All other tasks' response time and error rates were not normally distributed. Thus, Kruskal-Wallis and Mann-Whitney U tests were conducted for the non-normal data. Otherwise, we conducted a one-way ANOVA test on the normal data. Furthermore, Bonferroni correction was applied to reduce Type I errors (i.e., $p = .05/3$).

Maximum/Minimum Detection:

For the accuracy of maximum/minimum detection, we did not find any significant difference across conditions. However, for the response time in both maximum and minimum detection tasks, there was a significant difference between the three conditions (Figure 4.7). First, the result of the Kruskal-Wallis analysis for maximum detection was significantly different ($\chi 2 = 13.94, p < 0.001, df = 2$). To identify the location of the effect, Mann-Whitney U tests were conducted. For the maximum detection task, there were significant effects between SF-LG and Non-Simplified ($U = 58.00, p < .001$) but no significant effect between PIP and Non-Simplified nor SF-LG and PIP was found (SF-LG; $Mdn = 5599ms$, PIP;

$Mdn = 7788ms$, Non-Simplified; $Mdn = 10326ms$). Altogether, in comparison to the Non-Simplified graph, SF-LG was able to improve the response time for maximum detection tasks; such improvement was not found with PIP, however.

Because response time for the minimum detection task was normally distributed, we employed a one-way ANOVA test. A significant effect was found among the three conditions, $F(2, 27) = 12.62, p < .001$. Tukey post-hoc tests revealed that there was a significant difference between SF-LG and Non-Simplified ($p < 0.001$; SF-LG; $M = 5847ms$, PIP; $M = 8254ms$, Non-Simplified; $M = 11096ms$). Thus, parallel to the maximum detection, SF-LG again yielded the improved response time compared to the Non-Simplified.

Value Detection:

For the accuracy in value detection we did not find any significant condition effects. However, a Kruskal-Wallis test revealed a significant effect for response time ($\chi 2 = 17.99, p < 0.001, df = 2$). A Mann-Whitney U test yielded a significant effect between SF-LG and the Non-Simplified condition ($U = 56.00, p < .001$), SF-LG and PIP ($U = 71.00, p = .009$), as well as PIP and Non-Simplified ($U = 70.00, p = .007$). The median response time was 4328ms with SF-LG, 6583ms with PIP, and 14106ms with Non-Simplified.

Value Comparison:

For both response time and accuracy, we did not find any significant condition effects.

Trend Detection:

A Kruskal-Wallis test found a significant effect among the three conditions in

response time ($\chi 2 = 10.4, p = 0.005, df = 2$). Furthermore, Mann-Whitney U tests found significant effects between SF-LG and Non-Simplified graphs ($U = 63.00, p < .001$; SF-LG; $Mdn = 4890ms$, Non-Simplified; $Mdn = 12166ms$). No statistically significant effects were found in accuracy.

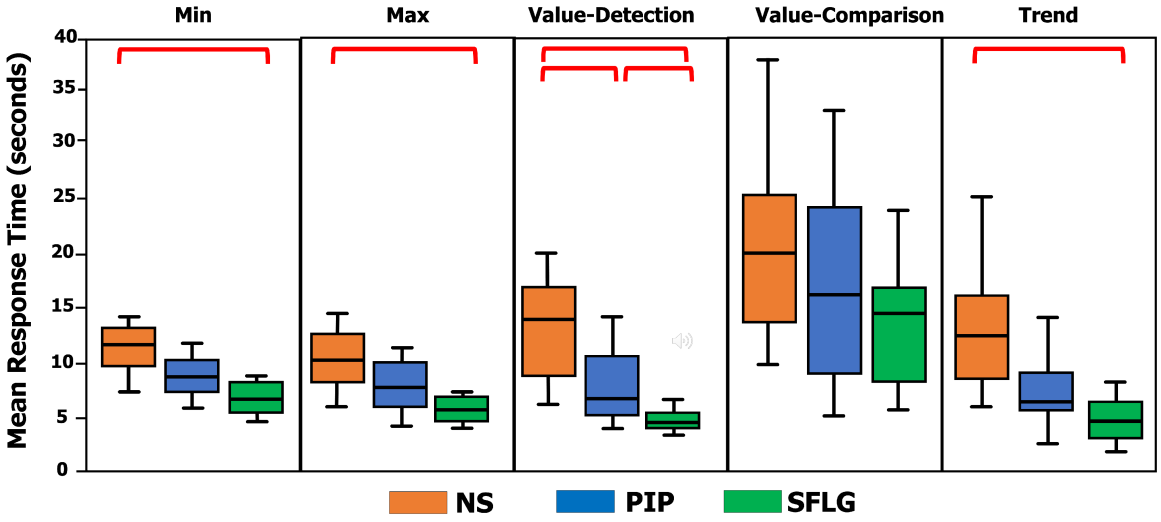


Figure 4.7: Mean response time (secs). Sig. differences between pairs are noted by the red lines.

Participant preference:

The majority of the participants (83.33%) preferred simplified graphs over Non-Simplified graphs, and more than half of the participants (53.33 %) ranked SF-LG the highest (Figure 4.8, Left chart). Many participants noted that the SF-LG graphs were more simple than the two other techniques and easier to understand. Some of the participants believed that representing many data points on a small smartwatch screen could backfire (e.g., P4: "Although more data point gives me the best data output, it is more uncomfortable and difficult to understand."[sic], P5: "[Non-Simplified] graph is too complex." [sic], P23: pointed out that "too many data points on the graph cause distraction and inaccurate interpretation").

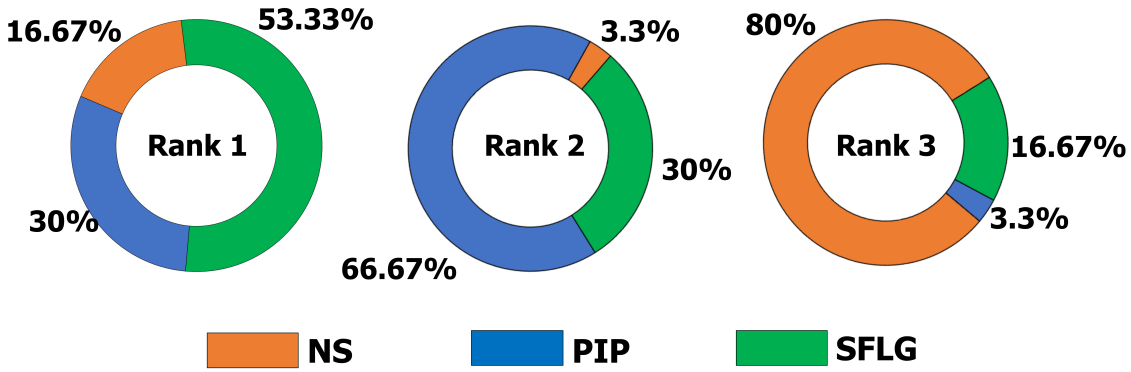


Figure 4.8: Percentage of each of the three techniques' rankings. SF-LG was ranked 1st by 53.33% (left), and ranked 2nd (in the middle) by 30%. We observe a higher number of 1st and 2nd rankings for SF-LG, than PIP or NS.

4.4 SPACE MAXIMIZATION AND EMBEDDED GRAPHS

SF-LG frees up space on the smartwatch display that we use to represent auxiliary information useful for queries requiring interlinked datasets. Collating data sources on multiple views is common and also available on commercial software applications. Tableau, for example, does this in the form of dashboards to aid in improved data analysis. To place additional charts on the available empty space on SF-LG, we develop a greedy algorithm consisting of two steps. First, the system calculates the entire space available. Second, we select regions for maximum graph legibility.

4.4.1 *Space calculation*

To calculate the available space around the line graph, we divide the graph into the area above and below. We explain our calculation for the lower area as the same method is applied to the upper area. The algorithm starts from the leftmost

data point of the graph. If the gradient of the line where this data point is located is positive, the algorithm calculates the horizontal line (parallel to the x-axis) connecting this data point to the first intersection of this horizontal line, and the line graph, or the border of the graph. The rectangular area beneath the calculated horizontal line will be the available space in the line graph which can be used for additional content. To continue, we just need to add a small value (e.g., one pixel) to the x-value of the starting data point, on the line graph, and repeat the process as for the first data point. For data points with a negative gradient, we have the same calculation in the upper area of the graph.

In another step, we calculate all available space under the line, and start from the bottom of the y-axis and move upward to reach the data point on the y-axis of the graph. Our algorithm does this by adding a small value, such as a pixel, to the y-value of the previous point. In each step we calculate the horizontal line, connecting this point and the intersection of this horizontal line with the graph. The output of this space calculation provides an array of available spaces with different heights, widths and areas. We use these calculated spaces in the next step of our greedy algorithm to allocate auxiliary visualizations. Algorithms 2 and 3 show how the available space beneath the line graph is computed. With few changes, this algorithm can be used to calculate all the available spaces above the line graph as well.

4.4.2 *Space allocation*

The space allocation component of the algorithm takes an array, ChartsArray, as one of the input arguments, representing the type of embedded visualization

we wish to add to the line graph (e.g., ChartsArray = [Bar, Pie, Pie, Bar, Line]). The output of the previously mentioned algorithm is another input argument containing all possible available spaces to present additional information in the line graph. For instance, ChartsArray = [Bar, Pie] indicates that we give the best available space on the screen to the first item, a bar chart. The pie chart uses the next largest available space. Charts, such as pie and donut are circular while graphs such as bar and line graphs have horizontal rectangular shapes. Since we calculate all available quadrilateral spaces in the line graph, the algorithm can use the available spaces that are horizontal-rectangular for line and bar charts, while using square-shaped spaces for pie and donut charts. As finding the exact square shape space is intractable, we define a square shape as a rectangle that follows one of the conditions in Formula 4.4.2. A pilot study on our smartwatch showed that rectangles that satisfy one of the following conditions could be considered as square spaces.

$$Height - (1/20)Height < Width < Height + (1/20)Height$$
$$Width - (1/20)Width < Height < Width + (1/20)Width$$

The previous steps lead to multiple small rectangular spaces. However, after experimental implementations, we noticed that such spaces are too small to represent many graph types such as line and bar charts. Similarly, available spaces with a very high (or low) ratio of width over height are not suitable for these graph types. Therefore, we implement restrictions on how we define horizontal rectangular space in our algorithm as follows: (i) the minimum height and the width of the space should be 60 pixels; (ii) the ratio of the width over height should be greater than $3/2$.

Based on the chart type to display and its order in ChartsArray, the algorithm seeks the largest available area and allocates that space around the central line graph. In the next step, the available spaces need to be recalculated only for those spaces that have a common area with the space occupied by the newly added chart. For instance, for ChartsArray = [Bar, Pie], the algorithm first tries to calculate all available spaces. Then it identifies the largest area with the aforementioned characteristics for horizontal-rectangular spaces to insert the bar graph. Finally, the algorithm recalculates the spaces that have a common area with the occupied space. The algorithm then runs a similar process to insert a pie chart. Changing the assignment priority gives a different layout (Figure 4.9). Algorithm 4 represents a pseudocode of the space allocation algorithm.

Algorithmus 2 : SpaceCalculation(P)

Input : A set P, of all data points in the line graph with x and y coordinates

Output : An array of all available spaces underneath the line graph (AllEmptySpacesArray)

```
1  YLeftTop = 0;
2  AllEmptySpacesArray[];
3  while YLeftTop! = p_0y do
4      PointIntersection = CalculateIntersectionPointWithGraph(0, YLeftTop);
5      AllEmptySpacesArray.add(Rectangular space between(0, YLeftTop,
        PointIntersection_X, PointIntersection_Y);
6      YLeftTop++;
7  for each data point t on the line segment between all P_i and P_i+1 do
8      PointIntersection = CalculateIntersectionPointWithGraph(t_X, t_Y);
9      AllEmptySpacesArray.add(Rectangular space
       _between(t_X, t_Y, PointIntersection_X, PointIntersection_Y));
10 return AllEmptySpacesArray
```

Algorithmus 3 : CalculateIntersectionPointWithGraph(X,Y)

Input : X and Y coordinates of a point

Output : Horizontal intersection of this point with graph

1 **for** *all data points of line graph with x-values greater than X* **do**

2 find the first two subsequent data points on the line graph, PNext and PPrevious, such that $PNext_Y <= Y and PPrevious_Y >= Y$;

3 **return** *the x and y coordinates of the horizontal intersection point between (X,Y) and the line segment between PPrevious and PNext*;

4 **if** *there are no such PNext and PPrevious* **then**

5 **return** *x coordinate of the right border of the line graph and Y*

Algorithmus 4 : GreedySpaceAllocation(ChartsArray[], AllSpaces[])

Input : An array of all available rectangular spaces sorted from maximum to minimum area (AllSpaces) and an array indicating the priority of representing charts (ChartsArray)

Output : Additional charts allocated to the optimal spaces within the main line graph

```
1  foreach CurrentChart in ChartArray do
2      if CurrentChart = Bar | CurrentChart = Line then
3          foreach CurrentSpace in AllSpaces do
4              if CurrentSpace = Rectangle then
5                  Embed the new Line\Bar chart in this space;
6                  SpaceCalculation(); \\to update AllSpaces
7                  break;
8      else
9          foreach CurrentSpace in AllSpaces do
10             if CurrentSpace = Square then
11                 Embed the new Pie\Donut chart in this space;
12                 SpaceCalculation(); \\to update AllSpaces
13                 break;
```

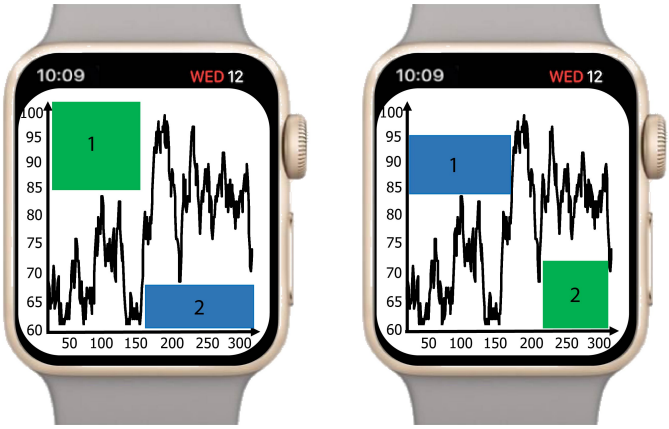


Figure 4.9: The greedy algorithm is provided with desired charts in a prioritized list of importance to be shown first. Based on the chart properties (rectangular or square) the algorithm identifies the optimal region to insert them. (Left) The priority given to the algorithm is to first find the largest square space and then a rectangular region (for example, a donut chart and bar chart respectively). (Right) The reversed priority (bar chart first, then donut chart) provides a different layout as it seeks to give the rectangular component the larger space first and then allocates space to the donut chart.

4.5 STUDY 2: EMBEDDED GRAPHS

Study 2 explored the capability of the SF-LG technique to efficiently present multiple chart contents on a smartwatch display to facilitate users' complex visual queries.

4.5.1 *Conditions and stimuli*

In Study 2, we evaluated the use of SF-LG against embedded graphs in a Non-Simplified graph. By comparing these two conditions, we will identify if generated spaces and embedded auxiliary information in SF-LG are effective or not compared to Non-Simplified graphs. We also included Flicking in our comparison as it is the most common method for exploring multiple graphs on a smartwatch. Our

investigation of the existing smartwatch applications (section 3.2) revealed that more than 90% of these applications use only one visualization in each visual display. A similar result also reported by Islam et al. [49]. Therefore including the Flicking condition is very similar to existing smartwatch applications to represent multiple charts.

One potential condition in this experiment could be dividing the screen into segments, for instance, on the y- or x-axis, and use each segment to represent one graph. As we wanted to use different types and shapes of graphs in this study (square and rectangle shapes), representing multiple graphs with different shapes could be highly dependent on the layout of these graphs, which will add more complexity to our study. So, we excluded this condition in this experiment. Thus, we arrive to three conditions; (i) SF-LG, (ii) Non-Simplified Embedding (NSE), and (iii) Flicking.

For the SF-LG and NSE conditions, we applied our greedy algorithm to identify the best locations to place the embedded charts. In the Flicking condition, one graph was presented per screen and users flicked through three screens in total per trial. Participants were allowed to go back and forth between the screens. For embedded charts, three of the most commonly used types on smartwatches were selected: line chart, bar chart, and pie chart [12]. Note, specific values were not represented on embedded charts, as these are often provided to help users get an overview of the central graph instead of presenting specific numerical values (e.g., to identify when the maximum/minimum happened, or to detect a trend).

The main line graph represented real heart rate data (e.g., the black line graphs on Figure 4.10) which ranged between 60 and 90 bpm, taken over 24 hours. This range was divided into three sub-ranges: low (60-70 bpm), mid (70-80 bpm), and high (80-90bpm). The proportion of these ranges were represented with an

embedded bar or pie chart (e.g., Low in Green, Medium in Yellow, and High in Red in the bar chart in Figure 4.10). The embedded line chart was randomly generated with data points ranging between 0-200 (e.g., the purple line chart in Figure 4.10), representing the burnt calories associated with the heart rate over 24 hours.

Note, our greedy algorithm selected either a line graph or a bar chart when the conserved space was rectangular. In contrast, a pie chart was selected when the conserved space was a square shape in the main line graph.

4.5.2 *Task*

The main goal of this study was to explore the participants' capability to link information between the main line graph and embedded charts. This analytic task consisted of the user understanding the overall tendency on heart rate (represented by the main line graph), and to identify how it was associated with burnt calories (an embedded chart). This query type is commonly of interest to users of in-situ visualizations, such as fitness enthusiasts [5].

Users had to follow these four steps to successfully complete a trial. Step 1: identify the most frequently occurring heart-rate range from the embedded chart (bar or pie). In Figure 4.10 (left), the highest value in the bar chart consists of heart rates in the high range (80-90bpm). Step 2: identify the corresponding region on the central line graph. This is marked in red in Figure 4.10 (center). Step 3: based on the region in Step 2, use the second embedded chart (purple line graph in Figure 4.10) to make a visual estimation of the average burnt calories. This required users to estimate this average based on the selected region in Figure 4.10

(right). Note, as both line graphs represented data recorded for 24 hours, this identification process was performed by comparing the relative segments of the line graph with the embedded line chart, on the x-axis. Step 4: indicate whether the mean within this region falls above or below the middle line (dotted in Figure 4.10 right), by pressing the Up or Down arrow keys.

Figure 4.10: Illustration of Study 2's task. (Left) Users need to first find the maximum value in the embedded bar chart. (Middle) Using the previously found value, the user next has to find the corresponding data points in the line graph. (Right) Finally, users provide their best visual estimate of the average value of the corresponding data points in the lower auxiliary line graph. This is representative of a task such as wanting to know the maximum burnt calories during peak performance

4.5.3 *Study design and procedure*

To prevent possible learning effects and cognitive fatigue, we used a between-subject design with one factor, the Visualization Technique (SF-LG vs. NSE vs. Flicking). First, participants went through a practice session. They performed all the tasks until they felt comfortable. Importantly, participants were asked to perform all the tasks as quickly and as accurately as possible. If the incorrect

response was entered, the users repeated that task after completing all trials. They performed 20 trials per visual condition. As in Study 1, participants used a Bluetooth keyboard to prevent any confounds. We measured response time and accuracy. After they completed the graph exploration tasks, we provided sample stimuli from each of the Visual Techniques and asked them to rank the visualization techniques based on their preference. The study took approximately 30 minutes.

4.5.4 *Participants*

Twenty-seven (27) new participants volunteered ($F = 15; M = 12; Mage = 27.31$; 26 right-handed) from a local university. Participants received a $15 gift card as compensation.

4.5.4.1 *Results*

We applied the same statistical tests as in Study 1.

Response Time:

A Kruskal-Wallis test revealed a significant difference among three conditions ($\chi 2 = 20.63$, $p < 0.001$, $df = 2$). Further, Mann-Whitney U tests yielded significant effects between all the pairs (SF-LG vs. NSE, $U = 55.00$, $p = 0.006$; SF-LG vs. Flick, $U = 45.00$, $p < 0.001$, NSE vs. Flick, $U = 45.00$, $p < .001$, SF-LG; $Mdn = 9764$ms, NSE; $Mdn = 18903$ms, Flicking; $Mdn = 37106$ms) (Figure 4.11).

Accuracy:

Because accuracy was normally distributed, we applied one-way ANOVA along with Tukey post-hoc tests. An ANOVA test revealed a significant difference among the three conditions, $F(2, 24) = 6.82, p = .005$. Pos-hoc tests revealed only one significant difference: SF-LG was significantly different from Flicking ($p = 0.003$; SF-LG; $M = 57.7\%$, Flicking; $M = 40.3\%$). Altogether, accuracy results indicated an encouraging potential for SF-LG on a smartwatch display. Although overall accuracy rate might not appear sufficiently high, placing multiple graphs on one page yielded a generally higher accuracy than placing them on different pages (Figure 4.11).

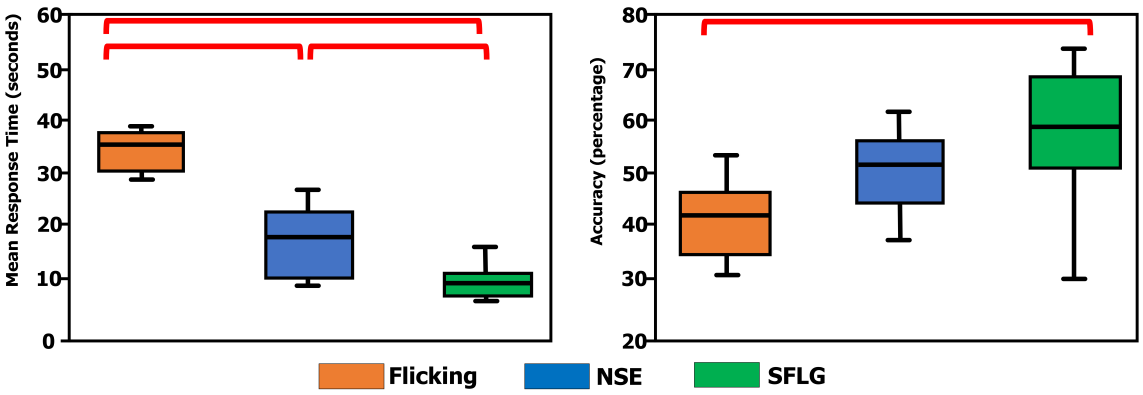


Figure 4.11: (Left) Participants' response time and (Right) Participants' accuracy in Study 2. significant differences between pairs are noted by the red lines.

Participants' preference:

More than 65% of the participants preferred the SF-LG method compared to the NSE or Flicking methods, and more than 60% of participants rated Flicking as the least favorable condition (Figure 4.12). Participants commented on the simplicity element of the SF-LG, P10: "the graph has less data points so that makes it less dense and easy to understand the trend."[sic]). Also, multiple participants noted that seeing all three graphs at once made the task easier and more accurate to

perform in comparison to flicking (e.g., P26: "I don't have to swipe to different screen to compare different conditions"[sic], P13: "I prefer the 'SF-LG' because [...] it is easier to see the relationship between all the graphs" [sic]).

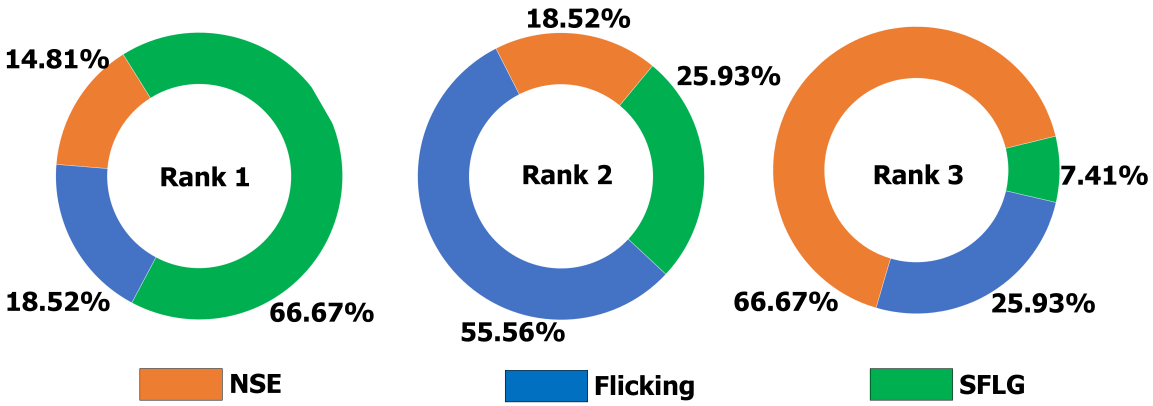


Figure 4.12: Percentage of each of the three techniques' rankings. SF-LG was ranked 1st by 66.67% of participants.

In summary, results from Study 2 reveal SF-LG benefits users on a smartwatch for tasks requiring viewers to relate information across multiple datasets. While we primarily tested a limited number of datasets, to facilitate three- or four-way comparisons across multiple datasets, additional mechanisms would be needed to include and remove graphs to be compared.

4.6 DISCUSSION

We briefly present potential SF-LG interaction techniques, use cases, as well as limitations and future work.

4.6.1 *SF-LG interaction techniques*

Through a few scenarios, we briefly present interaction techniques with SF-LG that resolve challenges with either the technique itself or with current ways of interacting with line graphs.

Details-on-Demand:

A user wants to view their heart rate at a certain point during their workout, which is one of the most basic smartwatch queries. To interact with their heart rate line graph, the user taps an area on the line graph. Additional charts, such as details of the simplification, or related bar or pie charts, fade-in to fill the largest spaces available. This approach can also be used to alleviate fat-finger interactions on a touch display [112], by providing the call-out of the details in a space made available by the SF-LG technique (Figure 4.13.a-c).

Multi-Level Details:

A user receives a weather warning notification, and tries to learn what the warning is about. The notification appears first as an icon in an available space. A tap on the notification icon opens a layer of details, into the largest area. Such multi-level details could be applied to other forms of information, such as receiving an email notification, and then viewing the content of the email either in an overlay or a call-out around the line graph (Figure 4.13.d-e).

Comparative Analysis:

A user wakes up and views their sleep quality data. They notice a spike in the data

at a certain time, and would like to know what caused the spike. We implemented an interaction method that allows a user to compare two data points using the embedded data. First, a user taps on the spike to view additional data (e.g., body temperature at the time of the spike). An auxiliary graph will fill in the largest available space. Next, they tap on another point in the sleep quality graph to view their body temperature at that time. A second embedded graph fills the second largest available space. This allows them to compare data at two points (Figure 4.13.f).

Managing Screen Updates:

An inherent challenge with the SF-LG is having to deal with changes in the view, as when a user flicks the line graph or when viewing streaming data. The challenge involves dealing with rapidly changing screen real-estate from one view to another. We addressed this concern in two steps.

In the first step, the embedded chart scales down but stays in the same position, while the central line graph is shifting out of view. This allows the embedded charts to remain in the same location within the line graph until such a point that the space is too small to display any content. As the embedded chart is shrinking in size, we only move it to the next available space if the point of interest on the line graph is still in view (Figure 4.13.g-i). If the selected point is no longer in view, the embedded chart disappears altogether.

4.6.2 *Takeaway lessons*

From our study results we propose several key takeaways:

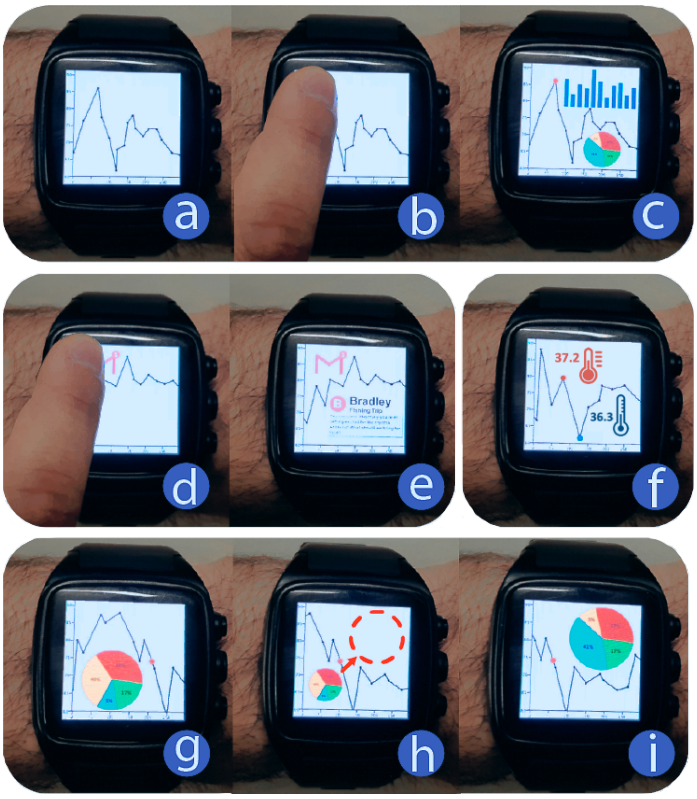


Figure 4.13: Scenario 1: (a) To obtain details-on-demand, (b) the user clicks a point in SF-LG. (c) This opens a callout in the most available region with details relevant to that point. Scenario 2: (d) A notification icon appears in the second most available region, and clicking it (e) opens details. Scenario 3: (f) Comparative analysis between two points on the line graph is possible. Scenario 4: (g) With streaming data, an embedded chart first (h) shrinks to the smallest size, before (i) moving to a region with sufficient space. This animated movement maintains visual consistency during the operation.

- Multilinked time-series data, collected and visualized on smartwatches, could be simplified using techniques that highlight a line graph's salient features.
- Simplifying a line graph frees up space, which importantly can be used to embed additional information.
- To present multiple charts that are linked to a line graph, having them on one view (via efficient space-filling) could yield better accuracy and response times than representing them on separate screens.

4.6.3 *Limitations*

Here, we present the limitations of the SF-LG approach and of our studies. First, we did not vary the simplification levels. We simplified PIP and SF-LG to a fixed number of points. As one potential technique to find the optimum window size, a preprocessing step could be added to SF-LG to calculate the number of high fluctuations for different window sizes. In a simplified line graph, if the difference of y-values of two consecutive data points is above a threshold (proportional to the range of all possible y-values) this can be considered as a height fluctuation. The window size with the minimum number of high fluctuations could be picked. We could have included such a factor in a larger study, which was beyond the focus of this initial work. Another limitation of our studies was the use of only a rectangular screen. While we believe SF-LG can be adapted to other display form factors (i.e. circular), we leave this for future work.

One of the advantages of SF-LG over PIP is that we can use it as a real-time algorithm for streaming data. However, we did not evaluate this element of SF-LG in practice. There are potential limitations of using SF-LG for streaming data. Depending on the shape of the simplified line graph, representing the streaming data, the scale and the location of auxiliary information could change continuously. We however demonstrate (described in SF-LG Interaction Techniques) how we can handle the continuously changing space available to embed graphs. With smooth animations and transitions, we believe we can alleviate these side effects of the space-filling method. Also, to prevent continuous quick location changing of the additional content, which could add cognitive load, a time threshold could be

added to maintain the location of auxiliary information for a specific minimum period of time.

Preventing extreme fluctuations in the line graph provides us with more suitable available spaces to add information to the line graph. Given that this is one of the main goals of SF-LG, for datasets that have periodic increasing and decreasing patterns in the data, if the size of windows in SF-LG is equal to periodic behavior of the data, this will cause a jagged simplified graph which is not suitable to present additional information, similar to PIP. This issue can be fixed by changing the size of the windows. In another particular case, if the values of most of the data points are close to the value of maximum/minimum data point (maximal/minimal points), the algorithm tends to select the data points that are far away from these maximal/minimal data points. This means that the maximum or minimum data points could be ignored in such a particular case.

Another limitation of our technique is that we did not provide labels on the axis of the auxiliary charts (e.g., bar and line chart) embedded in the main line graph. Although adding this information could help specific tasks, this may not be beneficial when the size of the supplemental chart is too small. This makes it hard for smartwatch users to read the labels and may add additional cognitive load. To solve this issue, the labels could be dynamically added to the auxiliary chart if its size is larger than a specific threshold to make sure labels are visible to the smartwatch users. Another essential factor in adding information to supplemental charts is the density and the number of data points in these charts. For instance, if a pie chart has only a few segments (e.g., two segments), depending on the size of the auxiliary pie chart, we can add values to the segments of this additional chart.

Choosing a suitable task, as we did for Study 2, is challenging. Our primary aim was to find a complex visual query, that users often reported needing based

on prior work [5, 100]. However, our choice of task resulted in a relatively low accuracy rate in Study 2. This is not surprising given that the task (i) required users to make a complex decision; (ii) involved viewing and comparing content across multiple small views; (iii) involved limited familiarity with these queries. However, simplifying the query could have resulted in floor effects, where we may not have observed differences among conditions. A suitable balance between task complexity and ecological relevance will need consideration for future tasks.

In future work, we intend on extending the Space-Filling techniques for glancing at more than one graph. Techniques such as Braided Charts [50] offer inspiration on how to handle multiple time-series graphs in one view. We plan to explore the effect of simplifying charts that include multiple time-series data. In such cases the simplification could be applied to one, multiple, or even all of the data sets in the graph based on the details needed by the users.

4.7 CONCLUSION

Line graphs, common on smartwatch applications, are one of the most frequently used visualizations to represent large time-series data. We presented SF-LG, a two-step technique, to make effective use of the smartwatch display space to assist users with in-situ visual queries. In the first step, SF-LG simplified a line graph, by focusing on salient features. We found that users can efficiently perceive simplified graphs to address basic queries. Furthermore, the simplification frees up space, allowing us in a second step to embed additional content around the line graph. In a second study, we found that embedding content around a simplified line graph can facilitate visual queries involving interlinked datasets. Finally,

we introduced interaction techniques that benefit from the SF-LG approach and proposed methods to improve in-situ analytics, directly on a smartwatch display.

5 BEZELGLIDE: BEZEL INTERACTION ON SMARTWATCHES

The goal of this study is to explore the amount of screen occlusion resulting from interacting with different areas of the smartwatch bezel. The results of this study enabled us to design an interaction technique with minimal screen occlusion.

5.1 STUDY 1: SMARTWATCH OCCLUSION

5.1.1 *Experimental Design*

In this experiment, we divided the smartwatch bezel into 24 equally sized segments. In each trial, one of these 24 segments was randomly highlighted in green and participants had to touch the target with their fingertip. To make sure that participants were pressing the correct target, we provided them with auditory and vibratory feedback. Thus, they had to try and hit the target until they heard and felt feedback. We segmented the bezel into 24 units (15 degrees each) based on the results of the optimal touch target size for smartwatches [43]. Using 24 segments provided sufficient granularity, as smaller segments would not be accurately interact-able.

We used a helmet with a mounted camera to capture images from the smartwatch screen. Any approach will create an offset, but we aimed to minimize this effect

based on prior methods used. Our camera setup was similar to that of prior work [113, 114] to measure screen occlusion. Vogel et al. showed that the captured images from the head mounted camera lineup closely with the users' line of sight (LOS) [114]. Similar to the approaches of Vogel et al. [113, 114], we ensured our mounted camera was centered with the eyes (Fig 2-left), without interfering with their LOS or disrupting them from performing the main task. In this experiment, participants had to perform the task three times for each segment. In total, we captured 3 (repetitions) × 24 (segments) × 11 (participants) = 792 images. A within-subject study design was used in this experiment and each session lasted approximately 30 minutes.

5.1.2 *Apparatus*

We used a smartwatch Galaxy Watch Active 2[1], which has a 44mm diameter display with a 360 pixel × 360 pixel resolution. We made sure that participants were not distracted during the study by disabling all notifications as well as the notification pane on the watch. To implement the experiment, we created a web app using HTML and JavaScript which was deployed natively on the smartwatch. Participants wore a helmet with a GoPro Hero3 camera mounted to the front side, to capture the images from the participants' view point, see Figure 5.1. We used these images to compute the degree of occlusion resulting from interacting with the watch bezel. We captured the images remotely for each trial via a smartphone remotely connected to the mounted camera. The research assistant conducting the

1 https://www.samsung.com/global/galaxy/galaxy-watch-active2/specs/

experiment captured stills on the smartphone as soon as the trial feedback was audible.

5.1.3 *Participants*

In total, we had 11 participants (9 males, 2 females, $M_{age} = 29$, SD = 5.39) complete the study. None of our participants were color blind as per the Ishihara color blind test[2] we administered. Participants were compensated with a $15 gift card for their time.

5.1.4 *Procedure*

This study was conducted during the COVID-19 pandemic. As such, special permission from the university ethics board was obtained for in-person human subjects data collection, and up-to-date health guidelines were strictly followed before, during, and after the experiment. Upon their arrival, participants were asked to follow these guidelines: pass a set of COVID-19 related health and travel screening questions, sanitize their hands, and wear a mask, see Figure 5.1. Participants were given a consent form to read, and a chance to ask questions about the study before signing the consent form.

During the study, participants were seated and wore the smartwatch on their left hand, interacting with their right index finger. We forced this aspect of the study (regardless of hand dominance) to avoid mixed results regarding the screen occlusion. This does not impact the results as screen occlusion characteristics

2 https://enchroma.com/pages/color-blind-test

would be mirrored across the vertical centre of the watch, and results are not reliant on time performance of participants. Participants were then given the helmet, with the camera mounted to it. We made sure the helmet was steady from the start and remained steady throughout the study, such that the camera did not move relative to the participants' point of view.

Participants were allowed to practice tapping the segments of the bezel until they felt comfortable, thus ensuring they could complete the task appropriately. We asked all participants to land on the targets as comfortably and naturally as possible. Participants were not instructed to perform fast since our focus was to capture the degree of screen occlusion. We did not inform participants of the experiment goal, and as such we did not ask them to minimize finger occlusion. Once participants were comfortable, we began the data capture phase. In this phase participants were randomly presented with a target and had to touch the target to end the trial. Each segment was presented as the target 3 times, for a total of 72 trials. There was a 1 second pause between each trial in which the bezel cleared during which the participant was instructed to move their finger away from the watch.

Figure 5.1: Screen occlusion study setup. A camera was mounted to the front side of a helmet to capture images of the finger on a region of the bezel from the user's vantage point. Images were then processed to compute the degree of occlusion.

5.1.5 *Image Processing*

To detect the amount of screen occlusion experienced by participants on the smartwatch when interacting with the bezel segments, an image processing algorithm was implemented in Python. We first cropped all the images to a fixed size such that the smartwatch screen was fully within the cropped region. Since the screen of the smartwatch was relatively bright and white, the visible screen region was extracted, converting the RGB image to a binary image and removing the noise using erode and dilation. To approximate the complete area of the screen, including the occluded part of the screen, the border of the screen was extracted. This was done by distinguishing the edges formed in the binary image between the skin from visible screen edges and the edges formed with the border of the smartwatch screen. These edges were computed using Canny edge detection, and for each edge pixel, the neighbourhood pixels in the RGB image were extracted. The edge pixels were identified to belong to the edge of the smartwatch screen if the red channel of the respective neighbourhood was not different from the average of the blue and green channels of the neighbourhood. This followed from the observation that the edges that the visible screen forms with the skin has a higher difference between the red channel and the other two channels. Once pixels belonging to the edge of the screen were identified, they were used as coordinates for the ellipse fitting algorithm [39]. Since the experiments were conducted under controlled conditions, with the data that was collected, the color between skin and the screen edge was distinguishable when the neighborhood of a pixel was considered. The algorithm and the parameters were defined specifically for this environment and

data collected. The results of the algorithm were verified by authors; ensuring the fitted ellipse matched the screen edge on a sub-sample of the complete dataset.

This ellipse fitting algorithm was then used because the screen would not always form a perfect circle on the camera from the users' perspective. The fitted ellipse model approximated the region of the binary image that would contain the complete screen of the smart watch. The ratio of the number of pixels in this region to the number of pixels representing the region of the visible screen on the binary image was used as an approximation of the region of the screen that was visible to the user. The ellipse-fitting algorithm [39] outlines this process. Figure 5.2 illustrates the result of our image processing on three of captured images in this study. Our image processing algorithm is provided in the Appendix.

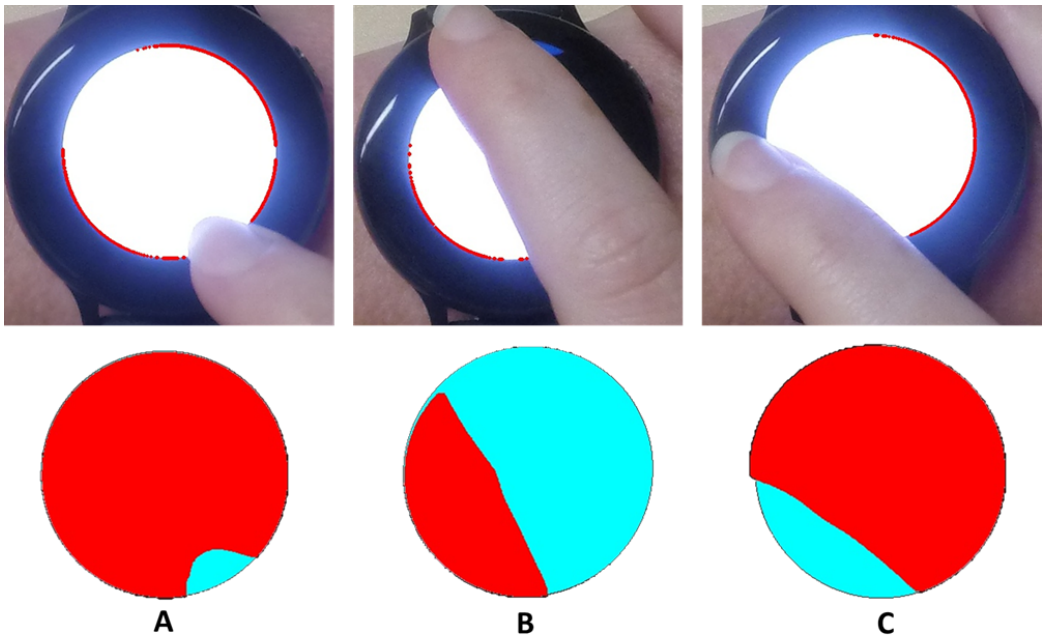


Figure 5.2: Using image processing techniques to calculate the screen visibility. A: 96%, B: 38% and C: 86% screen visibility.

5.1.6 *Results*

The average screen visibility across all segments, across all participants was 76%. The results of this experiment show that interacting with segments on the right lower corner of the screen provides the highest screen visibility, compared to other bezel segments. The indicated area in Figure 5.3 (denoted by the green arc inside the bezel segments) provides an average of 93.1% percent screen visibility, with all segments in that area providing over 90% visibility. This area ranges from 15° (just above 3:00 on a clock face) to 225° (7:30 on a clock face). The result also shows that interacting with segments on the left upper quadrant of the screen (from 105-175 degrees) has the worst screen occlusion, allowing only 51% of the screen to be visible, on average.

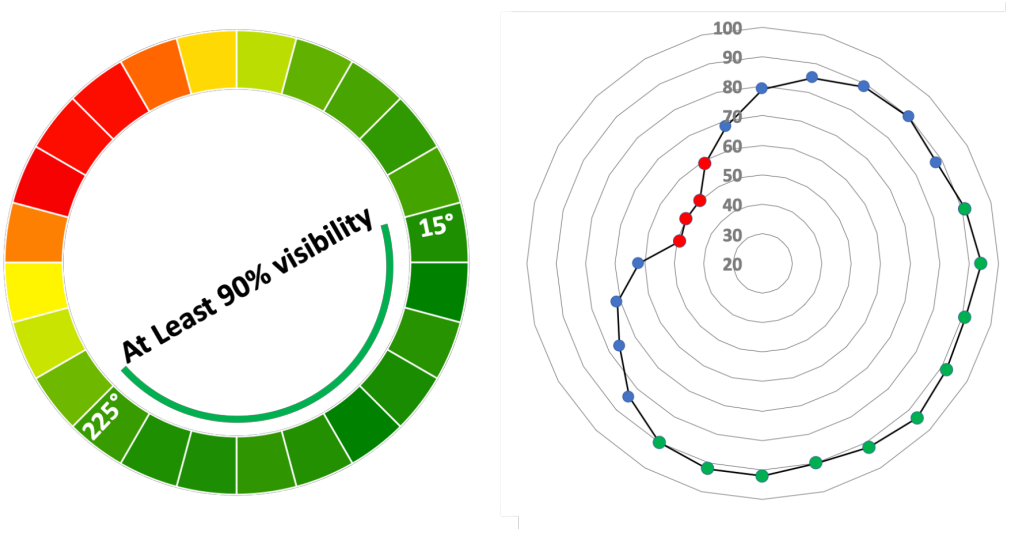


Figure 5.3: Left: heatmap from the screen occlusion experiment. The indicated area has screen visibility of at least 90% from 15° (2:30 on a clock face) to 225° (7:30 on a clock face). Right: spider graph representing the mean value of screen visibility for each segment in the screen occlusion study. The indicated bezel segments in green represent screen visibility of at least 90%.

5.1.7 *Discussion*

The average screen visibility across all segments was 76%, while the left upper quadrant is the area with the lowest screen visibility, and conversely the lower right quadrant provided the highest screen visibility. We note that using the entire bezel can be a good potential way to interact with the content on the screen due to the moderate overall average screen visibility. Moreover, utilizing the segments that provide the highest amount of screen visibility may also provide an interaction technique that is beneficial.

In this study we asked participants to wear their smartwatch on their left wrist to obtain consistent results. To generalize the results of this experiment for left handed smartwatch users (who may wear a smartwatch on the right wrist), we could mirror the results for each segment, flipped on the centre vertical position. Thus, the left lower corner would be the area with the least screen occlusion and the right upper quadrant would be the area with the most screen occlusion.

We note that while mobility may affect the performance of interacting with smart devices, this study did not look at this as a variable. Participants were given time, multiple chances, and the ability to move their finger on the touch screen until they were interacting with the highlighted segment. Thus, we captured the screen occlusion, which we expect would remain relatively similar across mobility conditions, such as while walking.

5.2 BEZEL INTERACTION TECHNIQUES

To overcome the fat finger and screen occlusion problems we designed BezelGlide; a novel suite of interaction techniques that can be utilized for data visualziations. Within this suite, we created Full BezelGlide and Partial BezelGlide. These were designed specifically to enable interaction with graphs, where being able to view the visual representation and associated content is essential. In this section, we describe how these techniques operate.

5.2.1 *Full BezelGlide*

From the occlusion study, we observed that across all bezel segments, an average of 76% of the screen was visible, with over 60% the bezel segments providing over 80% visibility and just under half providing over 90%. Inspired by existing bezel interaction techniques [2, 87, 128], Full Bezel Glide (FBG) is designed to utilize the *entire* bezel around the watch face to interact with data visualizations. This technique aims to address the 'fat finger' issue as it prevents the user from directly interacting with the content on the screen, as well as mitigates screen occlusion due to the bezel segments providing high screen visibility being included.

FBG is represented by a line, connecting the center of the screen (starting point of the line) to the position of the finger on the bezel (ending point of the line). By sliding the finger along the bezel, the location of the endpoint changes. As this line intersects with data points in the visualization, the value is displayed to the user; See Figure 5.4. Using the entire bezel allows the user to unambiguously access the entire screen area in a natural means.

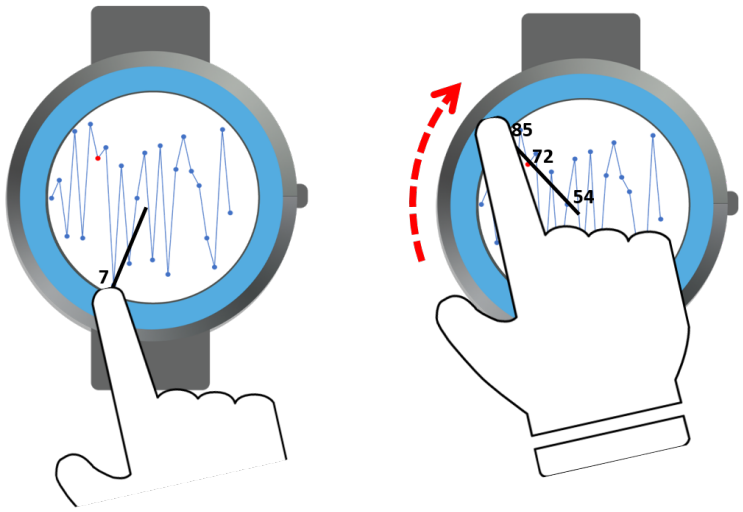


Figure 5.4: A demonstration of the FBG technique. Here the user can touch down anywhere on the bezel and a line will connect their touch location with the centre of the watch. The user can then slide their finger along the bezel, thus moving the line. Intersecting data points with the line will show the value.

5.2.2 *Partial BezelGlide*

We designed a second technique, Partial BezelGlide (PBG), to focus on the area of the bezel with at least 90% screen visibility, based on the results of the screen occlusion study. These results showed that interacting with the portion of the bezel between 15° (2:30 on a clock face) to 225° (7:30 on a clock face) ensures that at least 90% of the screen will be visible. Thus, the PBG technique only requires interaction to occur on this portion of the smartwatch bezel.

In PBG, we use the *interaction area* and the remaining area on the bezel, which we call the *projection area*; See Figure 5.5. The user does not interact within the projection area, they solely interacting with the interaction area. Similar to FBG, there is a line with start and end points. The starting point of the PBG line is the position of the finger on the interaction area. For each starting point, there is a mapping function that calculates the end point on the projection area of the bezel,

due to the reduced interaction area. The line behaves similarly to FBG, in that when the line intersects with data points the value is shown to the user.

Since we are using a smartwatch with a round watch face, first we need to convert the Cartesian coordinates (x and y) of the touch point on the interaction area to polar coordinates. The center of the polar coordinate, in our implementation, is the center of the display. To calculate the polar coordinates we need to calculate the distance of the point to the center as well as the angle of the line connecting this point and the center of the screen. We calculate the distance using the following formula:

$$r = \sqrt{(x - x_{center})^2 + (y - y_{center})^2}$$

We calculate the angle using the following equation:

$$\theta = \arctan \frac{y - y_{center}}{x - x_{center}}$$

After calculating r and θ, we have the polar coordinate of the start point of the line which is within the interaction area. A simple mapping function can then calculate the polar coordinate of the end point based on the polar coordinate of the start point. Notice that the interaction area covers 150° while the projection area covers 210° of the area of the smartwatch bezel. The following formula calculates the end point by scaling the start point from one range (interaction area on the bezel) to the new range (projection area on the bezel).

$$\theta_{Projection} = \frac{(\theta_{Interaction} - \theta_{OldRangeMin}) \times (NewRage)}{OldRange} + \theta_{NewRangeMin}$$

Notice that we just need to use the formula to calculate the angle, since the distance of the projected point to the center does not change. For the next step we calculate the Cartesian coordinates of the projected point.

Through this process, the interaction line will connect the starting point (touch point) and the calculated projected point on the projection area on the bezel. Similar to the previous technique, the intersection of the interaction line and data points will show the value of data points.

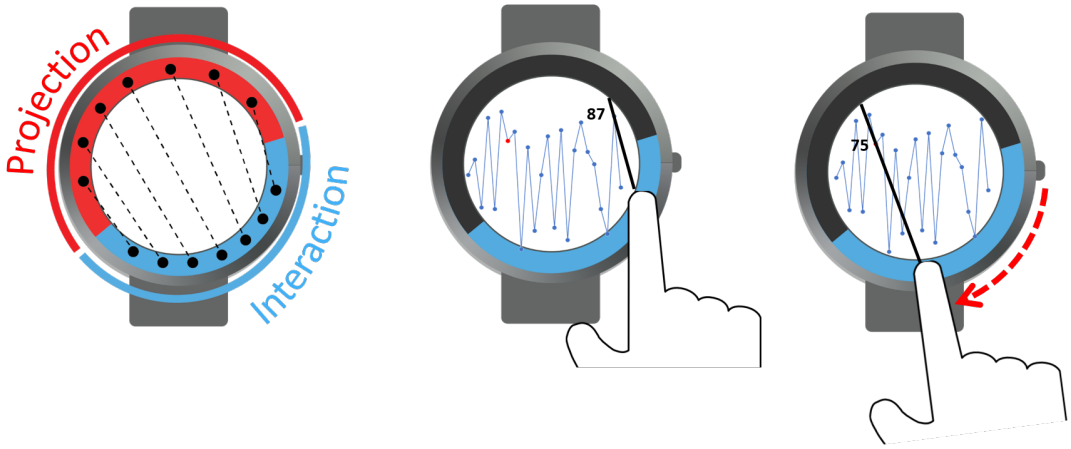


Figure 5.5: Left: shows how the mapping extends a touch point from the interaction space to the projection space. Middle and right: As the user glides their finger along the bezel the projection is updated and the intersecting points are selected on the graph.

5.3 STUDY 2: TECHNIQUE COMPARISON

Our second study investigates the performance of the two new techniques, comparing them with the baseline *Shift* technique [112]. Due to commonly interacting with smartwatches while on-the-go, we assessed our interaction techniques while standing (static) and walking. Ideally, interaction techniques on smartwatches can also generalize to mobility conditions [103].

5.3.1 *Shift Technique*

To evaluate our techniques we chose the Shift [112] technique as our baseline. It is a standard and well-known interaction technique designed to overcome the 'fat finger' effect on small displays. Using the Shift technique, the user can interact directly with the information on the screen by showing the content underneath the user's finger, in an offset miniature window. As the user moves their finger over the display, the offset window's position also changes to avoid occlusion. In our implementation of the Shift technique, the size of the offset window was 70×70 pixels. Pilot studies revealed that this size has optimal to display content.

To implement our Shift technique, we divided the screen into four smaller squares (top-left, top-right, bottom-left, and bottom-right segments). These segments' area is the same, with the same height and width. To interact with the data points on the bottom-left segment of the smartwatch display, the user tries to approximately put his finger as close as possible on the data pint. The offset window will appear on the top-left corner segment to show the content under the user's finger. The user then slides his finger on the screen to adjust his fingertip to see the data point's value in the offset window. Corresponding offset windows for the top-left, top-right and bottom-right segments are top-right, top-left, and top-right segments of the smartwatch, respectively. The offset windows' location was selected to minimize the screen occlusion that may happen by the index finger interacting with the smartwatch display. We also used zoom level three (3x) in the offset, as our pilot showed that this is the most effective zoom level for our implementation on the smartwatch.

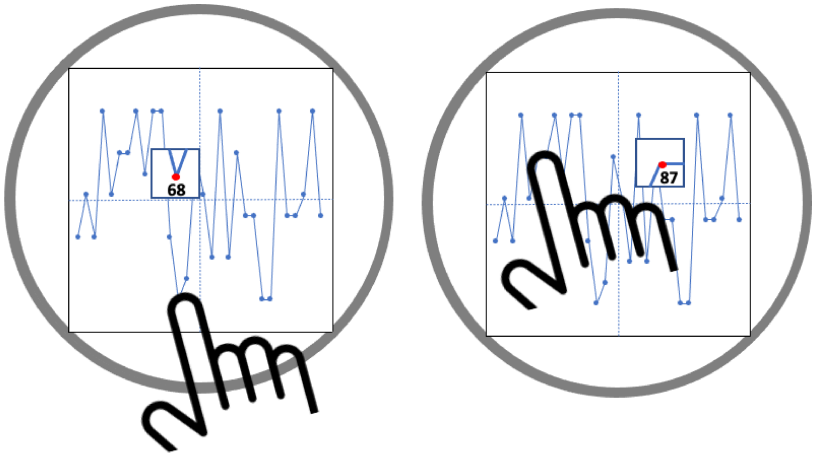


Figure 5.6: Left: Shows the shift techniques. The user is exploring the value of a data point in the bottom-left segment of the smartwatch. The offset window will be placed in the top-left segment. Right: interacting with the left-top segment puts the offset window on the top-right segment

.

We chose the Shift technique for a few reasons. 1) For increasingly dense data visualizations (e.g., graphs with pixel size data points), discrete tapping of data points becomes more difficult and requires additional steps each time a user wants to view a chart value. Thus, 2-step techniques do not afford continuous graph exploration, a highlight of our technique. 2) Shift was specifically designed for finger occlusion, and has shown promise on smartwatches [61, 99]. Shift has shown optimal performance under a number of contexts, including back-of-device input [44]. 3) Multi-stage tapping may not mitigate occlusion as the user would still be required to precisely tap each time. Continuous gestures, such as ours and Shift, afford the ability to adjust the finger position to quickly read out data values.

5.3.2 *Experimental Design*

For this experiment, we followed a 3x2 within-subjects study design, with the interaction technique (Shift, FBG, PBG) and mobility conditions (Standing and Walking) as factors. Evaluating PBG against FBG was necessary to identify if our consideration for occlusion mattered when browsing a graph. Although PBG is designed to minimize the 'fat finger' effect, it is not clear if the performance of FBG, as a more natural bezel interaction technique, is slower, faster or equal to PBG.

In this experiment, all participants performed all three interaction techniques in both mobility conditions, leading to 6 combinations in total. We counterbalanced only the order of the techniques (half of the participants performed FBG before PBG and vice versa). Participants started with the standing condition and then did the walking condition.

Stimulus:

We generated random line-graphs with 24 data points as the visualization in our study. We focused on the interaction of line-graphs as this technique is commonly used for health data [31, 74]. We used 24 data points as per prior smartwatch studies [12]. This number also follows the number of hours in a day to mimic data collected and represented at hourly intervals. A block (including one interaction and one mobility condition) consisted of 50 line-graphs, the same used across all participants, and randomly ordered for each participant. For each line-graph presented to the user, one of the 24 data points was randomly selected to be the target prior to the study. We used a range between 0 and 100 to generate data

point values. For all graphs, lines as well as data points are drawn in blue. We used red dots to highlight the target data point. We chose blue, red, as well as a white background, to ensure good contrast and legibility.

Task:

Data detection tasks are one of the most commonly used tasks across research exploring various data visualization techniques [45, 50, 60]. In this task, the main goal is to interact with the line-graph to detect the value of a highlighted data point, among all other data points. Selection occurs on a lift-off, and to cancel the user simply slides their finger past the 'active bezel' region to dismantle the intersecting line.

Participants:

We recruited 12 new participants (9 males, 3 females, $M_{age} = 22.2$, $SD = 5.62$) in this experiment. All of the participants were right handed and none of them were color blind as per the Ishihara color blind test we administered[3]. Participants were compensated with a $15 gift card for their time. None of the participants took part in Study 1.

Apparatus:

As in Study 1, we used the same Samsung Galaxy Watch Active 2 smartwatch. All three interaction techniques were implemented as web apps, by using HTML, JavaScript and CSS and which run native on the watch. All data and graphs were made in Microsoft Excel and exported for use in the implementation. Furthermore, to simulate the walking condition, a treadmill was used, see Figure 5.7.

3 https://enchroma.com/pages/color-blind-test

Figure 5.7: Apparatus of the study. Left: Standing condition; Right: Walking condition on a treadmill.

5.3.3 *Procedure*

Similarly to Study 1, this study was conducted during the COVID-19 pandemic. As such, proper health guidelines were strictly followed before, during, and after the experiment. Upon their arrival, participants were asked to follow these guidelines: answer COVID-19 related health and travel questions, sanitize their hands, and wear a mask; See Figure 5.7. After going over the ethics protocol with each participant, we provided the general information regarding the study.

First, we explained the interaction technique and we let participants practice until they felt comfortable to perform the task in that condition. Participants were informed that we were measuring their response time and we asked them to perform the task as fast as possible. We informed participants that they could take a break at the end of a block. At the end of the experiment, we asked participants

to fill out a questionnaire to share their opinions on and rank the three interaction techniques.

To start the trial, the participant tapped the smartwatch screen. Immediately a graph was shown and the timer would start. As soon as the participant reached the indicated data point in the graph, we flagged this event indicating the participant had actually crossed and seen the desired data point. Once this flag had been set, the time of the last touch up event stopped the timer and this was the response time for that trial. We did this so that participants could touch up and down as many times as they liked, while ensuring the desired data point had been crossed. Aside from response time, we had the participants tell us the value of the data point before moving to the next graph. This was done to validate that they had correctly seen the desired data point.

For the walking condition, we asked participants to walk on the treadmill and to adjust the speed to their Preferred Walking Speed (PWS) [103]. In addition to allowing for a safer experimental protocol, this ensured participants were comfortable with the mobility condition.

Lastly, we logged all participants interactions with the smartwatch display during each trial. This included touch points and the number of times they touched up and down. In total, we collected 3 (interaction techniques) $\times$ 2 (mobility conditions) $\times$ 50 (line-graphs) $\times$ 12 (participants) = 3600 trials.

5.3.4 *Results*

In this section, we report both quantitative and qualitative analyses from the data collected in the study.

Response Time:

For the data analysis, the results of a Shapiro-Wilk test showed that our data was not normally distributed. Therefore, Kruskal-Wallis and Mann-Whitney U tests were conducted to identify significant differences between interaction techniques. To reduce Type I error, Bonferroni correction was applied (i.e., $p = .05/3$).

The results of the Kruskal-Wallis test showed that there is a significant difference between three interaction conditions across the two mobility conditions ($\chi^2 = 597.62, p < 0.001, df = 2$). To identify the location of significant differences, we conducted Mann-Whitney U tests, and we found significant differences between all condition pairs: FBG and Shift ($U = 831936, p < 0.001$, Shift; Mdn = 2944ms, FBG; Mdn = 2148ms), PBG and Shift ($U = 941740, p < 0.001$, Shift; Mdn = 2944ms, PBG; Mdn = 1920ms) and PBG and FBG ($U = 770018, p < 0.001$, FBG; Mdn = 2148ms, PBG; Mdn = 1920ms). Figure 5.8 summaries the results.

We further analyzed response time for each interaction technique by dividing the watch-face into 24 even pie segments. We labeled each target in our graphs to belong to one of these 24 segments. We calculated the average response time for each technique for targets within each segment. Figure 5.9 (a) summarises the results. The average response time for targets in each segment is represented using a spider graph wherein the points closer to the center represent faster response times. We observe the following. We note that for PBG, the response time is the

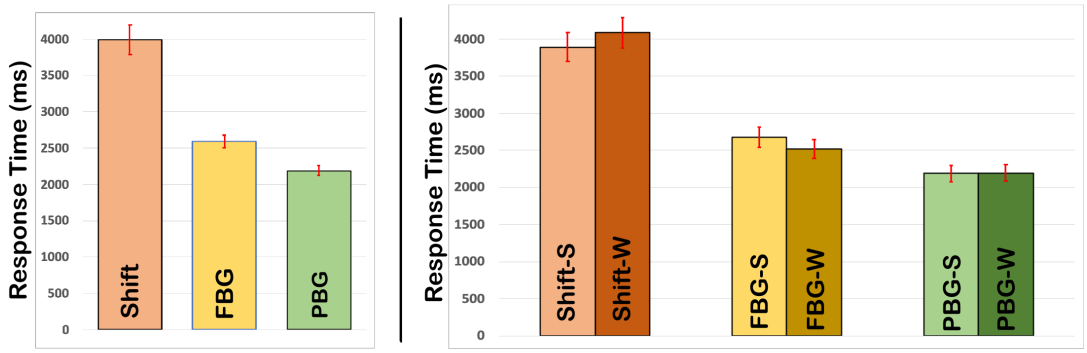


Figure 5.8: Left: Overall average response times (ms) for Shift, FBG, and PBG. Right: average response times for each interaction technique, separated out by mobility condition.

fastest for almost all segments. This demonstrates that PBG is the most efficient technique when targets appear in each of these pie segments on the smartwatch. The FBG technique also does well for some segments on the watch, but is slower for targets that appear near the right side (segments 23, 24 and 1-6). For the Shift technique, we note that while windowing mitigates the fat finger problem, response times are higher in the upper left corner. This may be due to the fact that the body of the finger still occludes the screen, and thus the window. In general, we observer larger fluctuations in response times around the different watch segments with the Shift and FBG techniques. In contrast, PBG shows the most consistent pattern.

We also classified the targets on the all line graphs into 4 concentric classes based on the location along the smartwatch radius. We calculated the average response time for each of these four classes for the different interaction techniques. For this comparison, we only consider PBG and FBG, as users clearly perform faster with these than the Shift technique. See Figure 5.9 (b). For both PBG and FBG, as we start from the center segment, and move towards the smartwatch edge, the response times increase. While these techniques utilize the bezel of the smartwatch,

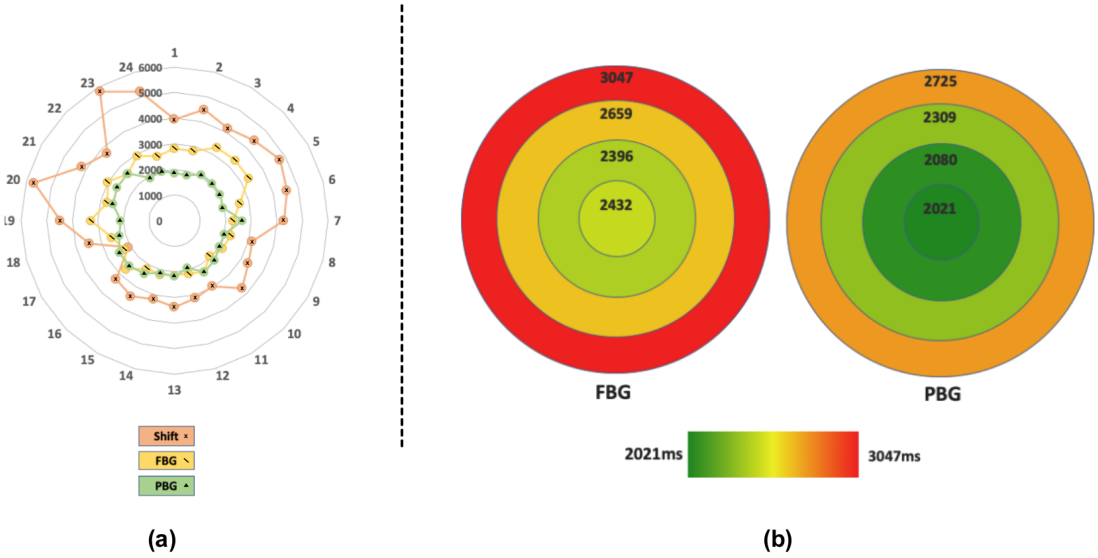


Figure 5.9: Response times divided by the location of the data point within pie segments (a) and concentric classes (b).

slight occlusion still occurs even in the highest visibility segments of the bezel. Thus, for points near the edge of the screen, these can still take slightly more time to acquire.

Touch Points:

To gain a better understanding of the interaction itself we further analyzed the touch points collected during the participant's interactions. We explored both touch down locations as well as the overall distance travelled during the interaction.

Touch Down Locations:

First, we wanted to explore the locations in which participants landed, and thus started their interaction from. Figure 5.10 (a) shows the results of the plot, with notable results for the FBG and PBG techniques. As can be seen, the PBG touch

down position is relatively condensed within the interaction area, although slightly skewed to the right. Surprisingly, the touch down locations for the FBG technique also tended to start in the bottom and bottom right bezel areas.

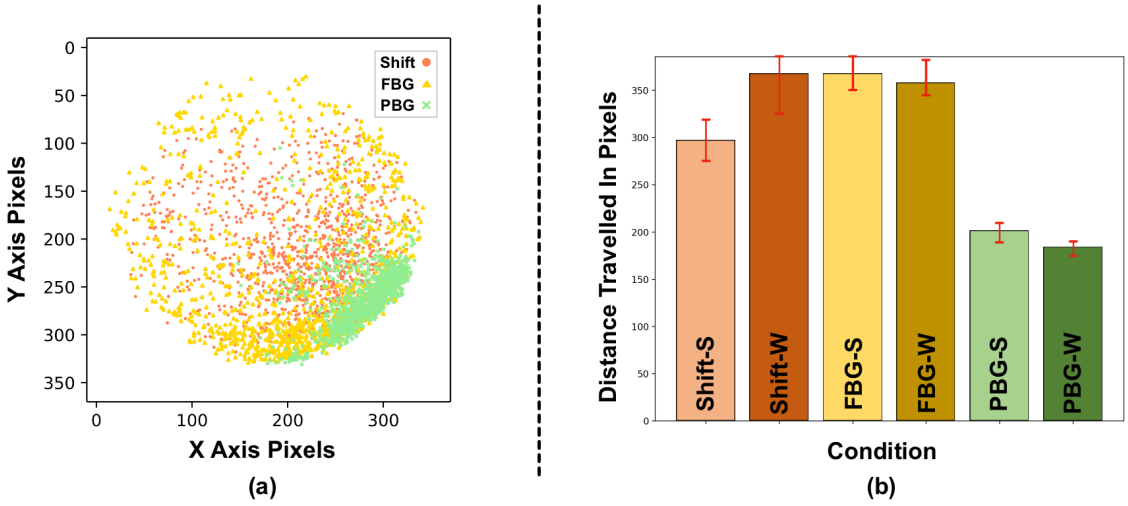


Figure 5.10: Touch point results for both (a) touch down locations and (b) average touch interaction distance travelled (with 95% confidence intervals denoted by the red bars). For touch distance travelled we note the circumference of the watch is 1131 pixels.

Touch Path Distance:

Due to the interesting results regarding the FBG touch down locations, we next aimed to explore the distance travelled during interactions by participants. The average distance travelled for each condition can be seen in Figure 5.10 (b). These show significantly reduced distances travelled for the only the PBG technique. We also note that mobility did not have an effect on this distance in both the BezelGlide interaction techniques.

5.3.5 *Mobility Conditions*

We were interested in seeing if there would be an interaction effect on response time between the two mobility conditions (standing and walking) and the three interaction techniques (Shift, FBG, and PBG). The result of our data analysis showed that there was no interaction effect. We also did not find any significant difference between the walking and running conditions for each interaction technique.

5.3.6 *Participant Preferences*

At the end of the experiment, participants filled out a questionnaire regarding their technique preference. We also asked them to provide reasons for liking or disliking a particular interaction techniques. 10 participants preferred the PBG technique compared to the two other interaction techniques (Shift and FBG).

Many participants pointed out that PBG is better than the other techniques because it fixed the screen occlusion problem. For instance, P1 mentioned *" my finger wasn't blocking the view and I could find the point easier."*., P7 commented *"Your hand is not in the way of seeing the number. "* and P10 noted that *"it only covers a tiny part of the screen so your fingers are not covering the whole screen as you move your hand around."*. With respect to screen occlusion, some participants mentioned that moving their head to adjust their view in the FBG technique made it harder to interact with, compared to PBG. For instance, P8 mentioned *"With FBG, sometimes your hand is in the way of the screen so you can't see the number."*.

Some participants mentioned that they liked having to move their finger less in the PBG technique to interact with graphs. For example, P2 pointed out that

"smaller area that interacts making it easier to get to a number ". Also, P11 mentioned that *"Limiting the area in my opinion made if faster to target the point and read the value and also the finger was not blocking the view."* As a disadvantage of PBG, some participants mentioned that it took time for them to get used to this technique.

5.3.7 *Discussion*

Our results show that both FBG and PBG are significantly faster than the Shift technique. We also showed that in PBG, by limiting the interaction area on the bezel to the area with maximum visibility, we decreased the response time and travel distance significantly compared to the FBG technique. We attribute this outcome to two effects. First, PBG reduces the overall amount of screen occlusion. In the Shift technique, participants had to directly interact with the screen, which blocked a great portion of the display. The FBG technique does not entirely mitigate screen occlusion concerns, especially while interacting with the left upper corner of the screen. This is resolved with PBG which allows for 90% of screen visibility, overall. Second, we observed the need for shorter displacements with PBG than with either FBG or Shift. FBG uses the entire bezel, and thus choosing the most appropriate landing point to make the intersecting segment appear can be time consuming. Further, in FBG participants tended to start in a position that allowed for them to view the graph before moving to the correct bezel location, and thus the data point desired; an already natural aspect of PBG. Shift uses the entire display, requiring the user to move significantly within this space.

5.4 DISCUSSION

We discuss our study 1 and 2 findings, and present the key take away lessons. We also discuss other potential applications of our techniques and present limitations of our work.

5.4.1 *Overall findings*

From study 1 we learn that we can achieve a high degree of screen visibility, if the interaction is limited to a region of the bezel in the lower right side (when interacting with the right hand). To retain 90% screen visibility, we should only use about 40% of the bezel region in the lower right quadrant (encompassing 15° to 225° clockwise). We also find that the opposite area, ie. the top left corner of the smartwatch bezel, has the least screen visibility. This area (105° to 175° counter-clockwise) has a screen visibility of 60% at most. We also found that in the worst case, interacting with the smartwatch bezel will yield a 50% screen visibility. Furthermore, on average across all segments, we obtain a 76% screen visibility. This leaves designers with at least 2/3 of the display to provide information. This makes the bezel a useful interactive region for mitigating screen occlusion on circular smartwatches.

In the second study, our results show that our participants performed better while they interacted with graphs using the PBG technique. PBG was designed based on the result of study 1 to mitigate the 'fat finger' and screen occlusion problems. On average, PBG was nearly 34% faster than the Shift techniques and 10% faster than FBG.

Further analysis shows that, for both PBG and FBG, if the target data point is close to the smartwatch bezel, it takes more time for users to reach and read the value of the data point. Whereas, if the data point is closer to the center of the screen, users can access them quicker. Figure 5.9 (b) shows that for all 4 segments (from center to the edge), PBG was more efficient compared to FBG.

Lastly, analyzing the touch points showed that total distance travelled was significantly reduced in the PBG technique. As well, both the BezelGlide techniques had users tending to start the interaction in or near the interaction area of PBG. Thus, the argument can be made that participants desired to start the interaction while being able to view and asses the data visualization before targeting the specific data point.

5.4.2 *Potential for Generalizability*

Our experiment used a common graph visualization, the line-graph. However, there are many other types of graph visualizations that a user may want to interact with on a smartwatch[5, 12]. In this section we illustrate how the PBG technique (which had the best performance in our experiment and causes the least amount of occlusion) can be generalized to other commonly used graph types on smartwatches.

Blascheck et al. [12] showed that donut, bar and radial bar charts are the most commonly used visualization techniques on smartwatches. We thus illustrate the potential of PBG with these charts. We also show how PBG generalizes to pie charts, as they are very similar to donut charts.

In PBG, a data point appears when the interaction segment intersects with the line graph. As the segment intersects multiple parts of the line graph, we show all the data points intersecting with this line. To generalize this interaction technique to other chart types, we must modify slightly how we display values. For donut and pie charts, this requires showing values for the segments of the chart where the interaction line crosses the outer edge of the chart (see Figure 5.11). For bar and radial bar charts, we utilize the top most edge of the individual bars. If the interaction line crosses this top edge, then we display the value of that bar, Figure 5.11.

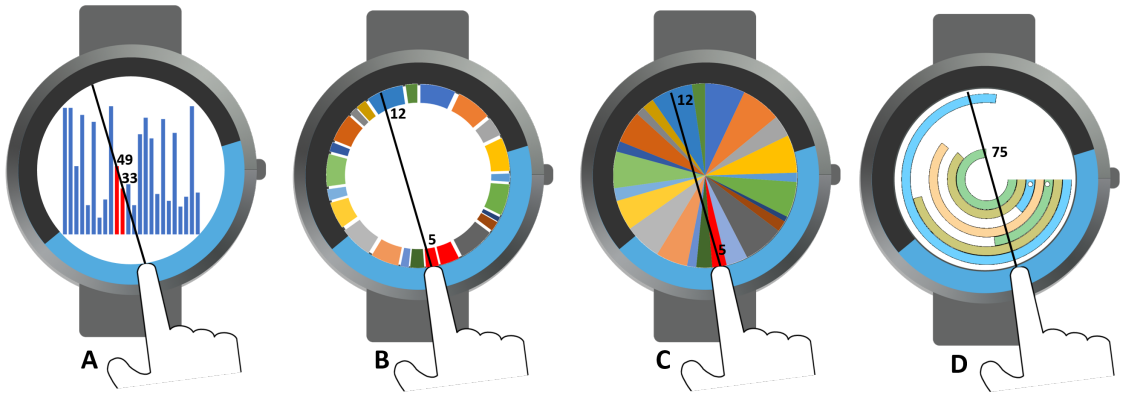


Figure 5.11: Using PBG on, A: Bar chart, B: Donut chart, C: Pie chart, and D: Radial bar chart

5.4.3 *Other Applications*

We focus primarily on graph selection. However, our BezelGlide techniques can also serve to interact with other content on smartwatches. For example, PBG (without the interaction line) can be used as an interaction tool for menu and list scrolling tasks on smartwatches. Currently, the most common way to scroll through menus on existing smartwatches is by swiping down or up through the

menu items. Some smartwatches have used a technique similar to FBG for menu scrolling, however through this work we note the reduced performance to that of PBG. This is due to the trade-off regarding clutching and screen occlusion that can still occur.

Zooming is a continuous interaction technique with which the user directly interacts with the content on the smartwatch display (e.g., map, or photo). Our PBG (with no interaction line) could also be used as a zoom-in and zoom-out interaction technique. The user can swipe to the right, using PBG to zoom-in and swipe left for zoom-out.

To make PBG a more general interaction tool, we can adapt the PBG functionality based on the display content. For instance, the user can use PBG to scroll through menu items to select a map application on the smartwatch. In the map application, PBG will change its functionality to zoom-in/-out. Similarly, on the map, the user may select a point of interest (such as a restaurant) and then PBG may change to navigate through images (e.g., next/previous image).

5.4.4 *Limitations and Future Work*

One of the limitations of this research is the minor difference between the camera angle and the participant's line of sight (LOS) in our first study. If the difference between the camera angle and LOS is large enough, the captured images would be very different from the participant's point of view, which may result in unreliable findings. Although there is no known practical way to solve this issue, a few papers proposed ways to minimize the difference between the camera angle and LOS [113, 114], to make sure the captured images are close to participants' point of

view. We followed their guidelines and camera setup to capture images from the participants' LOS performing the task in study 1. However, there will always be a minor offset so as not to block the participant's line of sight using this method.

Another limitation is that we developed our techniques based on wearing the smartwatch on the left wrist. While it does not affect FBG, PBG would need adjustments. We are assuming that we could mirror our interaction area and flip on the centre vertical position, however future work should explore it further. Also, for our study 2, we asked our participants to walk on a treadmill rather than walking outdoors, which is a more realistic situation. We did this to have more control on participants' speed and to prevent adding more complexity to our study. Further, our task in the second experiment was limited only to a data value detection task. Although, this is one of the most commonly used tasks across various visualization techniques, evaluating PBG and FBG with additional tasks, such as data comparison and minimum/maximum detection, could be useful. Finally, while focusing on the interaction techniques, we consciously left out additional UI contents and displayed only the line-graph trial. We are aware that UIs on smartwatches contains, along with graphs, text, icons, and other important information. While it was out of the scope of this paper, it would be interesting to analyze how PBG and FBG can fit along current UIs.

While we asked participants in Study 1 to interact as naturally as possible, in real-world scenarios users may interact not only with the index but possibly other fingers as well. Assessing smartwatch input occlusion in a contextual setting may show different results. Capturing such information can further the design of novel smartwatch techniques. Our current lab study findings are limited and future work is needed to test more ecologically valid settings, such as jogging or running

outdoors. Amini et al. also suggests a variety of tasks end-users require in such settings [5]. We aim to validate our interaction technique with these tasks as well.

5.4.5 *Take away lessons*

Based on the results of our studies, we propose the following key take aways:

- interacting with the entire smartwatch bezel can mitigate the 'fat finger' problem, but does not entirely resolve the screen occlusion issue.
- interacting with the right lower corner of the smartwatch display, within a specific angle range (15° - 255° clockwise) is the most suitable place to minimize screen occlusion on smartwatch displays.
- using this optimal area on the smartwatch, our BezelGlide technique can facilitate effective value detection tasks on line graphs, in static and mobile conditions.

5.5 CONCLUSION

We present BezelGlide, a suite of interactive smartwatch techniques to facilitate interacting with graphs on a small display. Our technique is motivated by the need to inspect graph contents, in-situ, during the activity generating the data, such as a while running. BezelGlide is designed to minimize occlusion with the display and thus allows using the smartwatch bezel as an input modality. First, we examined regions around the bezel that minimize screen occlusion. We find that bezel regions in the lower right quadrant have optimal screen visibility of at least

90%. This led us to design two BezelGlide techniques: Full BezelGlide (FBG) and Partial BezelGlide (PBG). The FBG technique involves gliding the finger around the watch's entire bezel to target values using an intersecting segment from the screen centre to the fingertip. PBG uses a non-linear transformation of points from a specific bezel region, chosen from our optimal screen visibility results in our first study, to cover the entire screen with a similar segment. We find that PBG, enables rapid interaction with line-graphs, both while standing and walking. PBG also outperforms both FBG and Shift, a technique to mitigate touchscreen occlusion and the 'fat finger' problem, in a value detection task on line graphs. We demonstrate that PBG can be generalized to other graph types, including donut-, bar-, pie-, and radial-bar charts on smartwatches. In this work, we offer one of the first input modalities allowing for improved interaction with graphs on smartwatches. This furthers the ability to explore data visualizations, all while considering minimal display occlusion on small displays.

6 EDGESELECT: SMARTWATCH DATA INTERACTION WITH MINIMAL SCREEN OCCLUSION

In this chapter, we examine the possibility of minimizing the interactive region of a smartwatch display, to mitigate screen occlusion while exploring multiple graphs. This small interactive region, is designed to occupy less than 10 percent of the smartwatch display without hindering the density of graphs displayed on the watch. Our two main objectives summarized as our contributions include: 1) quantifying the screen occlusion resulting from interacting with the entire smartwatch display; and 2) taking advantage of segments with minimum screen occlusion to design an interaction technique to interact with multiple graphs.

6.1 STUDY 1: MEASURING SCREEN OCCLUSION OF SMARTWATCH DISPLAYS

The main goal of this study is to measure the screen visibility of the smartwatch display while participants are interacting with the entire smartwatch display. The difference between this study and the screen visibility study in the previous chapter 5.1, is the interaction area. In BezelGlide, we only focus on interacting with the smartwatch bezel. However, our screen occlusion experiment is further generalized to assess the degree of screen occlusion when interacting with the entire smartwatch display, including the screen bezel.

6.1.1 *Experimental Design*

In this experiment, we divide the screen display into 88 targets. The location and the number of segments on the right/left and up/down half of the display was the same (symmetric). We picked 88 targets, as this was the largest number of targets we could include, without making touch cumbersome. Using a larger number of subdivisions would reduce the number of targets and make it tedious to select each cell. The system highlighted a cell or target (red colour) on each trial, randomly selected from 88 targets. Only the highlighted target was visible to the participants, and they had to hit the target to be able to move on to the next target. To make sure they hit the target, we provided them with auditory and visual feedback, changing the background colour from white to light green after landing on the target. We also asked them to hold their finger on the target for three seconds to ensure that our camera recorded the display and their finger position on the smartwatch display. In this experiment, participants had to select each of the 88 targets once (in random order), then they had a two-minute break, and they had to repeat this cycle two more times. So in total, each participant selected each of the 88 targets three (x3) times. Increasing the number of samples would result in a more accurate result but needed to be balanced with the total overall time to collect the data. To measure the screen occlusion, we placed a camera on a head strap designed to hold an active camera at the center of the participant's forehead. Unlike the work done in previous chapter 5, we decided to record videos as this makes the recording process easier. To minimize the offsets between the camera lens angle and participant's point-of-view, we followed the same guidelines explained by previous papers [73, 114] by placing the camera in

the middle of the forehead and as close as possible to the eyes without interfering with the participant's eyesight and affecting their performance on the task.

6.1.2 *Apparatus*

We used a GoPro Hero 9 camera to record the experiment. This action camera can capture high-quality videos with a built-in stabilizer, which ensures clear videos for image/video processing.

6.1.3 *Participants*

We recruited 13 participants, all students from a local university (5 female, 8 male, $M_{age} = 21.29, SD = 1.5$). Participants were all right-handed, and none were colour blind (based on Ishihara color blindness test). Please note that, although we used auditory and visual feedback, we did not test our participants' auditory impairment in this experiment. Because the primary goal of this experiment was not to measure the response time and performance of our participants, it was enough to provide them with visual feedback. Because we measured the colour blindness of our participants and made sure they were not colour blind, using the auditory feedback was an extra level of feedback.

6.1.4 *Procedures*

This study was conducted during the COVID-19 pandemic, so we followed the guidelines of the university ethics board to conduct the study safely for both the

participants and the researchers. At the beginning of each experiment session, the researcher ensured that the smartwatch and all devices, including the camera and head strap, were appropriately sanitized. We also asked all of our participants to answer a set of health-related questions to make sure they did not have COVID-related symptoms. We also made sure that they did not travel outside of the province for the past two weeks. Most importantly, we tried to minimize direct interaction with participants during the experiment to maintain social distancing.

Upon arrival, and after following COVID-related health guidelines, we explained the primary goal of the experiment to our participants, and asked them to read and sign the consent form to start the experiment. To maintain consistency, we asked our participants to wear the smartwatch on their left hand and interact with the smartwatch display with their right index finger. This kept the image processing simple and consistent across all participants.

In the next step, we asked our participants to wear the head strap with the mounted camera. We ensured the strap was comfortable but also tightly secured. The camera was placed at the center of the participant's forehead and toward the smartwatch display to capture the interaction with the screen following the guidelines from previous papers [73, 114].

6.1.5 *Video Processing*

We used a similar approach 5, to detect the occluded area of the smartwatch. In this section, we describe the steps taken that are different from the previous chapter 6 that addresses some of the limitations of that approach. In lieu of capturing static images of the participant interacting with the screen, a video of

the complete session was recorded. Note that the screen turns green when the participant touches the targets and remains so for 3 seconds. This screen colour change is used to synchronize the events generated on the smartwatch with the recorded video. Extracting multiple frames for each touch event from the video allows removing frames that have artifacts such as motion blur by comparing the calculations from a series of frames and removing the frames that are outliers. Choosing green (Figure 6.2a & Figure 6.2b) as the colour for the background also helps with differentiating the edge pixels of the smartwatch screen from the edge pixels of the finger. To further differentiate the two types of edge pixels, the calculated visible screen area was made black (Figure 6.2d) prior to calculating which type each edge pixel is. This ensures the edge pixels of the smartwatch (red edge in Figure 6.2b) have much lower values on the red channel since these pixels will be surrounded mainly by black pixels. While the edge pixels of the finger (blue edge in Figure 6.2b) would have more pixels with higher values on the red channel. These adjustments allowed the parameters of image processing to be less sensitive and therefore more robust across participants and trials. Another issue with the previous approach was the different lighting conditions and location of the smartwatch on a given frame. The former was rectified by more closely managing the lighting of the environment and allowing time for the camera's auto colour balance to stabilize. The latter was caused by how different participants hold their hands when interacting with the smartwatch. Compared to asking the participant to hold their hand in a particular position or using a stand to ensure the same posture is used by different participants, capturing the occlusion under natural interactions is beneficial to the following analysis as it would be more representative of how participants would interact with the smartwatch in the real world. An assumption we make in designing this study is that for a given

hand posture, the occlusion by moving only the head with the gaze fixed on the smartwatch would not vary much. To reduce the manual intervention needed to calculate the position of the smartwatch on each frame, fiducial markers were used. A frame for the smartwatch containing the markers was designed and attached to the smartwatch (see Figure 6.2a). The frame was placed such that the markers would always be on the opposite side from which the participants would interact with the screen. The frame essentially extends outward in one direction, and the markers were attached on the frame such that they were aligned to the direction in which the frame is being extended. Since the actual size of the markers was known, an offset from a given corner of any marker to the center of the screen can also be calculated. Since the markers are placed so that they are aligned to the direction in which the frame is extending, and the watch is circular, this offset would remain the same for any orientation in which the frame is attached. These measurements could then be translated to pixels, which would inform the center of the screen on the frame as well as provide a scale used to calculate the size of the image to crop from the frame prior to performing the occlusion calculations (the blue square on Figure 6.2a represents the cropping region on the original frame, Figure 6.2b-d are frames cropped from Figure 6.2a). Multiple different markers are placed on the frame to improve robustness. As the tracking of the fiducial markers ignores frames that are not clear, it also functions as another measure to discard frames on top of the outliers being calculated as described above.

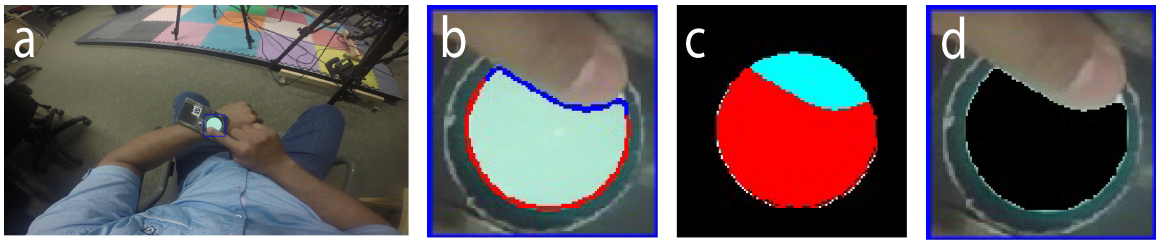


Figure 6.1: a) Frame as captured by the camera. The blue rectangle represents the region that will be cropped for further processing based on the fiducial markers, b) Extracted edges of the screen colour-coded based on the type: Red for edge pixels of the smartwatch, Blue for edge pixels of the finger, c) Fitted ellipse and occluded region, and d) Image used to calculate the edge pixel type. The detected screen region is replaced by black.

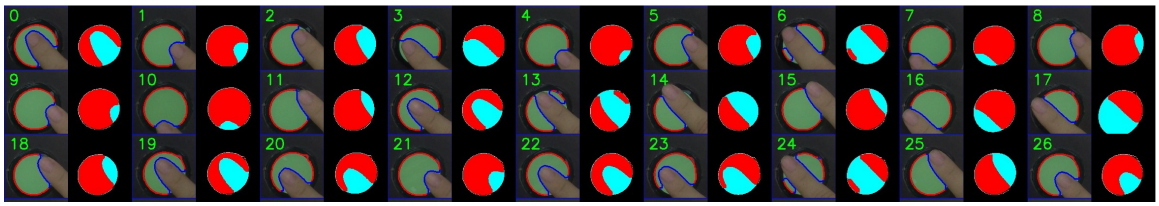


Figure 6.2: The result of our video/image processing algorithm for one of the participants. Samples 1,2,5,8,9,10 show that the interaction with segments on the right, bottom and bottom-right side of the smartwatch display have minimum screen occlusion. On the other hand, samples 0,6,13,14,19,and 24 show that interaction with segments on the top-left corner of the smartwatch leads to the highest screen occlusion levels

6.1.6 *Results*

Figure 6.4 illustrates the results of the screen occlusion study in the form of a heatmap. Our analysis shows that interaction with the segments on the top-left quarter of the smartwatch display has the worst screen visibility with an average of 44.35% screen visibility compared to 74.22%, 73.31%, and 86.04% for bottom-left, top-right and bottom-right quarters respectively. In addition, 96% and 35% are the best and worst screen visibility, which belong to the top-left and bottom-right quarters of the smartwatch display, respectively. Figure 6.4 shows that interacting with the outermost segments (smartwatch bezel) on the right, bottom, and bottom-

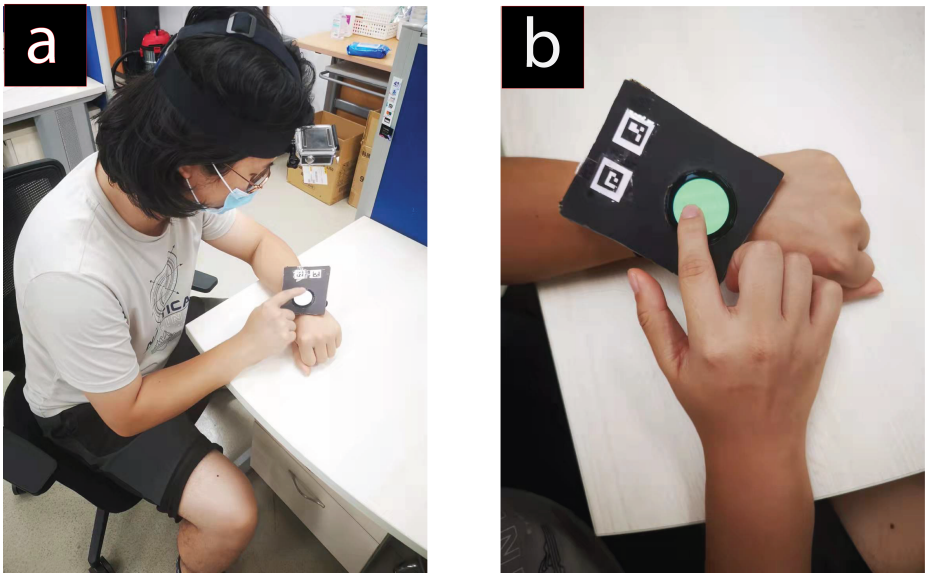


Figure 6.3: Study setup. a) this image shows the placement of head strap and the camera in Study 1. Participants had to put their wrist on the table to prevent fatigue. b) shows the location and orientation of QR indicators that were used in our image/video processing. Participants had to hold their finger on the target for 3 seconds after hitting the target.

right side of the smartwatch display provide the best screen visibility, with a minimum of 89% and a maximum of 96% screen visibility. Segments on the smartwatch bezel on the top-left corner of the display provide the least screen visibility with a minimum of 38% and a maximum of 53% screen visibility.

Although segments of the smartwatch bezel on the right, bottom-right and bottom side of the display have the best screen visibility, adjacent segments to the smartwatch bezel (inner segments on the smartwatch display) have high screen visibility as well, making them potentially good options for an interactive region to facilitate explore graphs on the smartwatch display.

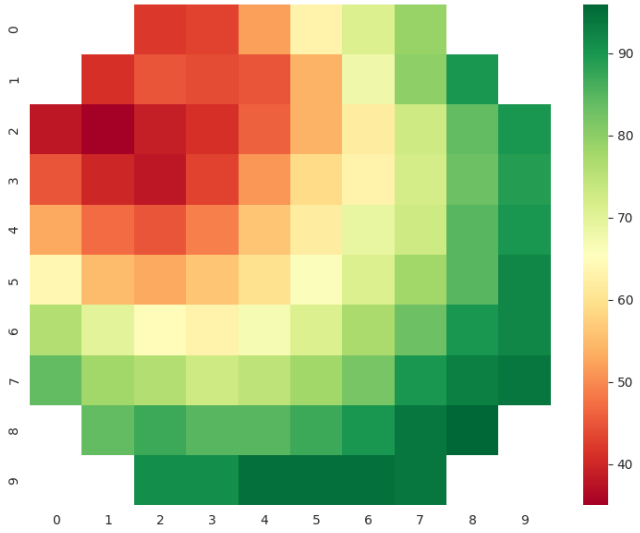


Figure 6.4: Screen visibility of smartwatch display while participants were interacting with different segments. The number on each segment represents the average percentage of screen visibility when participants hit the target on that specific segment of the display. This visualization was made by using colours from red (minimum screen visibility, 35%), yellow (median of screen visibility, 65%) and green (maximum screen visibility, 95%) colour gradient.

6.1.7 *Discussion*

While earlier related work examined screen occlusion when interacting with the smartwatch bezel [73], we offer a broader set of results to include the entire smartwatch display. Our analysis confirmed the result of screen occlusion 5, showing that interaction with the outermost segments (smartwatch bezel) on the right, bottom-right and bottom side of the smartwatch display ensures the highest level of screen visibility. Interacting with the top left corner of the smartwatch bezel provides least screen visibility. Our results suggest that we can use a sufficiently large region, taking less than 10% of the entire screen space to interact with all contents on the display. This provides us an opportunity to facilitate interacting

with multiple graphs using minimal finger movement around the display. We next demonstrate how we can use our selected regions with more than 90% screen visibility to interact with graphs on the small smartwatch display.

6.2 INTERACTION TECHNIQUE

This section will introduce EdgeSelect, an interaction technique to interact with multiple graphs on small screens of smartwatches with minimal screen occlusion. The result of the first experiment revealed how interaction with different segments of the smartwatch could affect the screen visibility from the user's perspective. The first row of Figure 6.5 shows all the segments of the smartwatch with at least 90%, 85% and 80% screen visibility. To design our interaction technique, we will only focus on the segments with at least 85% and 90% screen visibility. We will call these two areas, Large and Small interaction areas, respectively, throughout this paper. We excluded segments with at least 80% screen visibility because they occupy more than 10% of the smartwatch display, which means the interaction area is significantly larger than two other interaction areas.

The border of selected segments for both Large and Small interaction areas is jagged (Figure 6.5 first row). This makes it hard and less intuitive for the smartwatch users to move and keep their fingers within the interaction area. Because of this, we smoothed the border of the interaction area for both the Large and Small interaction areas (Figure 6.5 second row). The curved shape of the interaction area helps the smartwatch users to move their fingers along the curved display bezel, making the interaction more intuitive for smartwatches with circular displays.

Unlike BezelGlid, the main goal of EdgeSelect is to take advantage of the width of the interaction area as a selection tool to select one of the multiple graphs on the smartwatch display. Therefore, by dividing the width of the interaction area into a number of bands (Figure 6.5 second row), each band can be used to interact with one graph. Our pilot study showed that three is the optimum number of bands for the interaction area in both Large and Small interaction areas. The Inner-band is the layer that is closest to the center of the display. The Middle-band is the layer in the middle and the Outer-band is the layer closest to the smartwatch bezel.

Each of the three bands of the interaction area can be used to interact with one graph. A smartwatch user can start interacting with a specific graph by hitting the correct band of the interaction area. As soon as the user interacts with a band, the outer border of the corresponding graph will be highlighted to show the selected graph. If the user hits a wrong band, the user can slide his/her finger to reach the correct band. Sliding the finger within a specific band will allow the user to interact with the data points of the corresponding graph, see Figure 6.6.

6.3 STUDY 2: SIZE OF THE INTERACTION AREA AND THE NUMBER OF DATA POINTS

The size of the interaction area in EdgeSelect is inversely proportional to the size of the visible area for displaying content on the screen. Thus Study 2 involved a series of target acquisition tasks designed to evaluate participants' performance while interacting with two different sizes of interaction areas: Small (area with screen visibility of 90% or higher) and Large (area with screen visibility of 85% or higher). We measured how each of the three interaction area bands could be used

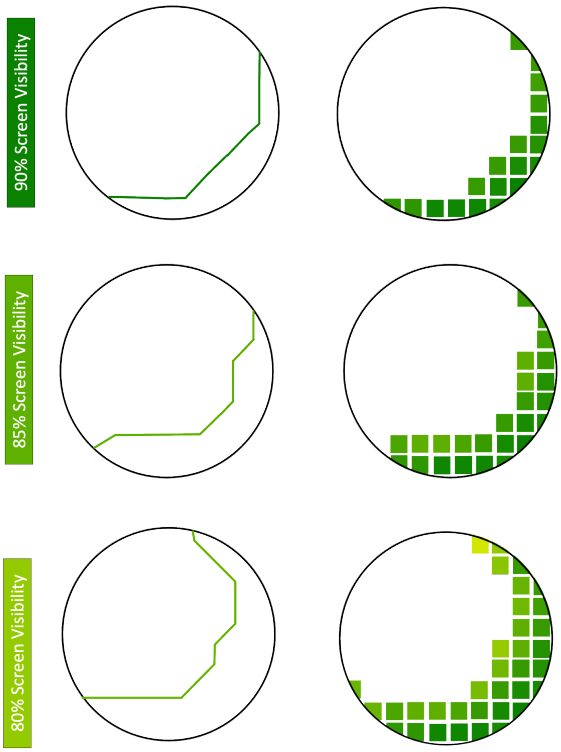


Figure 6.5: First row: represents all the segments of the smartwatch display with at least 90%, 85% and 80% screen visibility. Second row: three different interaction areas corresponding to each screen visibility level. The width of the interaction area is divided into three bands. Each band will be dedicated to interact with one graph

to interact with graphs of differing densities, in terms of target selection time, for each size of the interaction region. We also measured the maximum number of data points per graph that users can effectively interact with using EdgeSelect.

In order to measure the maximum number of data points, we used three linear arrays of targets (Figure 6.7) instead of actual graphs. This enabled us to have precise control over the number and position of targets for target acquisition measurement. Each array represents a graph, and the targets represent graph data points.

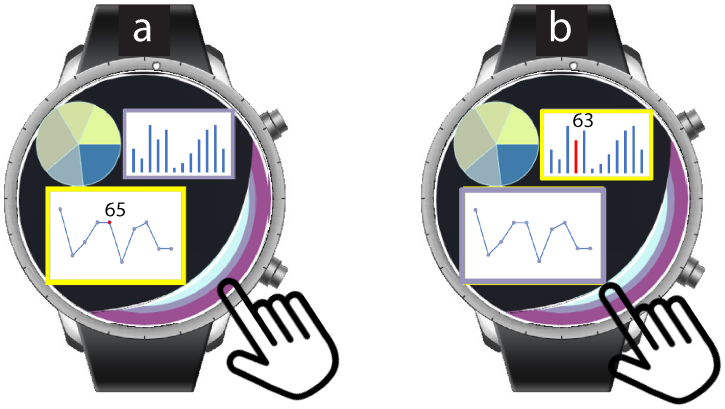


Figure 6.6: a) Interacting with line graph using the Outer-band of interaction area b) Sliding the index finger into the Middle-band to interact with bar chart. Moving from one band to another band changes the highlighted graph to make it easier for the user to identify the graph they are interacting with

6.3.1 *Experimental Task*

In order to determine the maximum number of points a user can interact with on a single graph, we varied the number of points on the graphs dynamically, depending on the success or failure of the participant. **Selection was triggered upon finger lift-off.** If a participant was able to select the specific highlighted target on a graph within a specified time limit (5 seconds), we considered that trial a success, and increased the number of points in that graph. A failed trial occurred when the participant failed to select the highlighter point in the time allotted or if the wrong iem was selected. In the event of a failed trial we decreased the number of points in that graph. We adapted the graphs to generate a point randomly in four segments of the graph before increasing or decreasing the number of points (Figure 6.7). Generating targets in 4 segments forced the participant to select a point in 4 different regions of the interaction area, one far to the left, two in the

middle, and one to the far right. This ensured that every part of the interaction area was tested. In general, as the number of points increased, the difficulty in selecting the point increased. Over a series of trials, the participant would converge on an optimal maximum number of points in each graph, failing when the graph is too dense, and succeeding when the graph was below or at the optimal maximum density. If the participant selected targets successfully in 3 or more segments, the number of points in the graph would be increased. Otherwise, the number of points would be decreased (Figure 6.7). The experiment ended if the number of points decreased 3 times to the same level. For example, if the participant was seeing 10 points, failed target acquisition twice, and the number of points decreased to 9, then succeeded at target acquisition 3 times, and increased the number of points back to 10, but then failed again, causing the number of points to decrease to 9 twice more (total of 3 times decreasing to 9), the experiment would end.

6.3.2 *Participants*

We recruited 12 (new) participants (8 males and 4 females, $M_{age} = 22.08, SD = 3.50$) in Study 2. Our participants were all right-handed and used their watches on their left wrist. Four of these participants used smartwatches regularly, two occasionally, and four rarely used smartwatches.

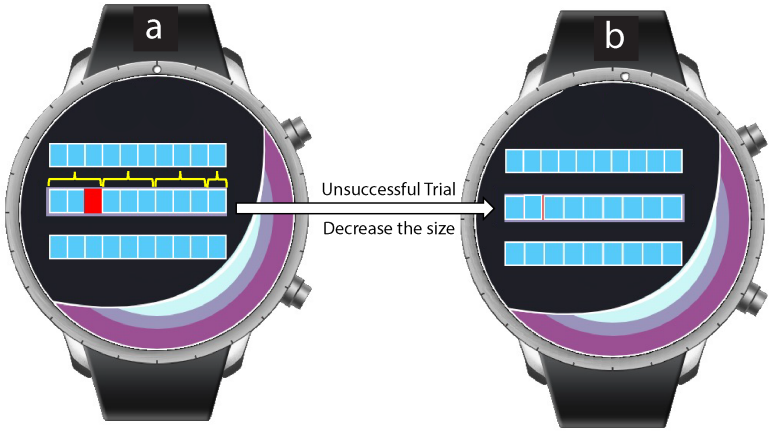


Figure 6.7: a) This image represents three arrays of targets. Our participants had to hit the highlighted target. In this experiment, the Inner-band interacts with the top array, the Middle-band interacts with the middle array, and the Outer-band interacts with the bottom array. To ensure participants interact with all the segments of each band, we divided the arrays into 4 major segments and picked one random target from each segment (indicated by yellow brackets in the middle array). b) If participants successfully perform the target selection task, we increased the number of targets in that specific array.

6.3.3 *Apparatus*

As in Study 1, we used the Samsung Galaxy Watch Active 2 smartwatch. The smartwatch-based interaction software was implemented a a web app, using HTML, JavaScript, and CSS and ran natively on the watch.

6.3.4 *Procedure*

Similar to the previous experiment, we followed all COVID-related health guidelines to ensure the safety of participants and researchers. After participants arrived, we explained the objectives of the experiment. After signing the consent form, one of the researchers explained the experiment task and study progression to the participants. They were asked to practice with the interaction area and three

bands to hit the random targets for as long as needed to feel comfortable (Figure 6.7). They were told that accuracy and response time were the two critical measurements, so they had to perform as quickly *and* as accurately as possible. We had two main conditions in this study: Large and Small interaction areas and, accordingly, different sizes of bands (Figure 6.5). We used a within-subject design, so all participants interacted with both the Large and Small interaction areas.

Similar to the first study (Figure 6.3), we asked our participants to put their hands on a table to prevent fatigue. They could also have a break between trials if needed. When they finished the experiment, we asked them to answer some open-ended questions regarding interacting with EdgeSelect and using different bands.

6.3.5 *Results*

In this section we describe the outcome of this experiment.

6.3.5.1 *Response Time*

The result of a Shapiro-Wilk test revealed that response time data was not normally distributed ($p < 0.05$). Accordingly, to identify the significant differences between conditions, Friedman and Wilcoxon signed-ranked tests were conducted. Bonferroni correction was applied to minimize Type 1 error (i.e., $p = .05/3$).

The result of the Wilcoxon test showed that there was a significant difference between the the Large and Small interaction areas (Figure 6.8-left, $p < 0.001$, Large; $Mdn = 1788$ms, Small; $Mdn = 1754$ms). This shows that the smaller interaction area results in a faster response time. Our further data analysis showed that there

was a significant difference between Large and Small interaction areas for the Inner-band condition (Figure 6.8-middle, $p < 0.002$, Large-Inner-band; $Mdn = 1884$ms, Small-Inner-band; $Mdn = 1795$ms), Middle-band (Figure 6.8-middle, $p = 0.038$, Large-Middle-band; $Mdn = 1734$ms, Small-Middle-band; $Mdn = 1765$ms), and the Outer-band (Figure 6.8-middle, $p < 0.001$, Large-Outer-band; $Mdn = 1769$ms, Small-Outer-band; $Mdn = 1682$ms). The result of Friedman test showed that there was not a significant difference between Inner-, Middle- and Outer-bands across both Large and Small interaction areas Figure 6.8-right.

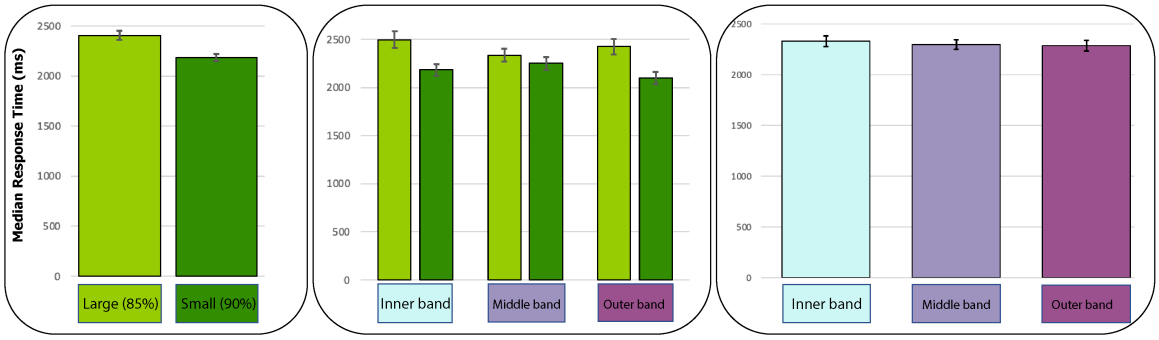


Figure 6.8: Response time result
Comparing the response time of participants interacting with Large and Small interaction areas (left), and different bands of interaction area (right). The middle chart shows the performance of participants interacting with all combinations of bands and interaction areas. The error bars represent 95% confidence interval.

6.3.5.2 *Number of Points*

In this section, we will analyze the general pattern, as well as the minimum and the maximum number of points our participants reached using EdgeSelect. We will analyze how the size of the interaction area and each of the three bands could affect the participants' performance.

Our data analysis showed that there is a significant difference between the Large and Small interaction area across all trials (Figure 6.9-left, $p < 0.001$, Large-

overall; Mdn = 10 points, Small-overall; Mdn = 11 points). Deeper analysis showed that there is a significant difference between the minimum number of points our participants reached, between the Large and Small interaction areas (Figure 6.9-left, $p < 0.001$, Large-min; Mdn = 7 points, Small-min; Mdn = 9 points). A similar result captured for maximum number of points our participants achieved, between two Large and Small conditions (Figure 6.9-left, $p < 0.001$, Large-max; Mdn = 11 data points, Small-max; Mdn = 12 data points). The result from a Friedman test revealed that there was a significant difference between the Inner-band, Middle-band, and Outer-band in terms of number of points. By comparing the conditions we observed a significant difference between Inner-band and Middle-band conditions (Figure 6.9-right, $p < 0.001$, Inner-band; Mdn = 10 points, Middle-band; Mdn = 11 points) and between Middle-band and Outer-band conditions (Figure 6.9-right, $p = 0.02$, Outer-band; Mdn = 10 points, Middle-band; Mdn = 11 points)

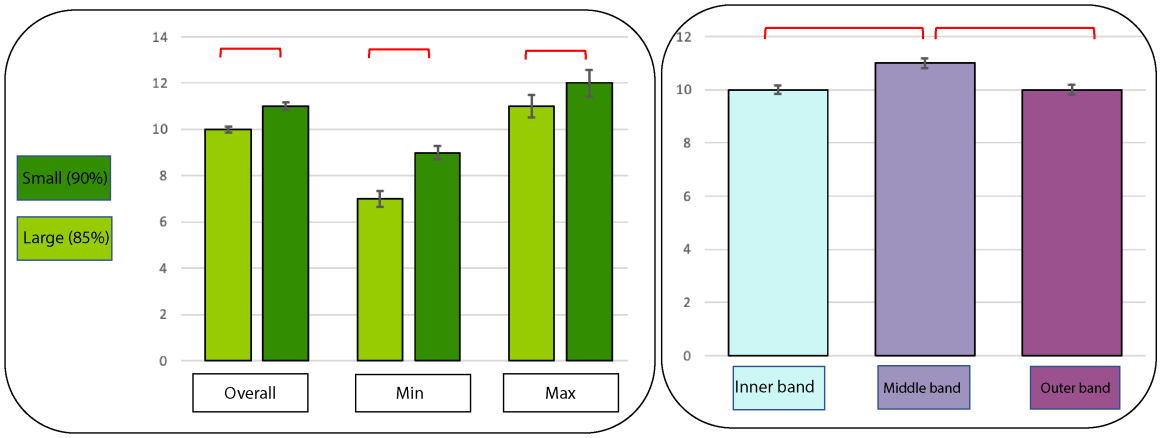


Figure 6.9: Left: Overall average, as well as the minimum and maximum number of points participants could select in this experiment using the Small and Large interaction areas. Right: Average number of points participants were able to select in each band across both Large and Small interaction areas. (red brackets indicate pairwise significance)

Figure 6.10 shows all the consecutive (successful and unsuccessful) trials across all study participants, with each graph representing trials in a different interaction band. Each separately-coloured line represents the trials of one participant until they reached the end of the experiment. The y- and x-axis of all three graphs are scaled to 25 and 50 respectively, making it easier to compare the existing patterns. Comparing these three graphs shows that in general, trials in the Outer-band terminated faster than trials in the Inner-band and Middle-band (Figure 6.10). In addition, participants took longer to finish the trials from the Middle-band compared to other bands.

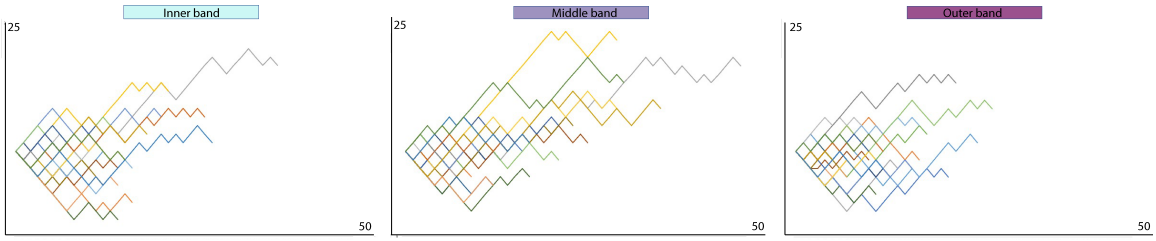


Figure 6.10: Number of points (y-axis) of all trials (x-axis) for each participant (each line).

6.3.5.3 *Failed Trials*

In this section, we will analyze how many times participants failed the trials. A failed trial occurred either when the user lifted their finger from the incorrect target, or when the 5 seconds elapsed without the user selecting a target. Our analysis showed that participants significantly failed more in the Large interaction area compared to the Small interaction area ($p < 0.001$, Large; $Mdn = 9$ fails, Small; $Mdn = 7$ fails). We also found that the number of failures was significantly different only between the Inner-band and Middle-band ($p < 0.038$, Inner-band; $Mdn = 7$ fails, Small; $Mdn = 9$ fails) across both Large and Small conditions.

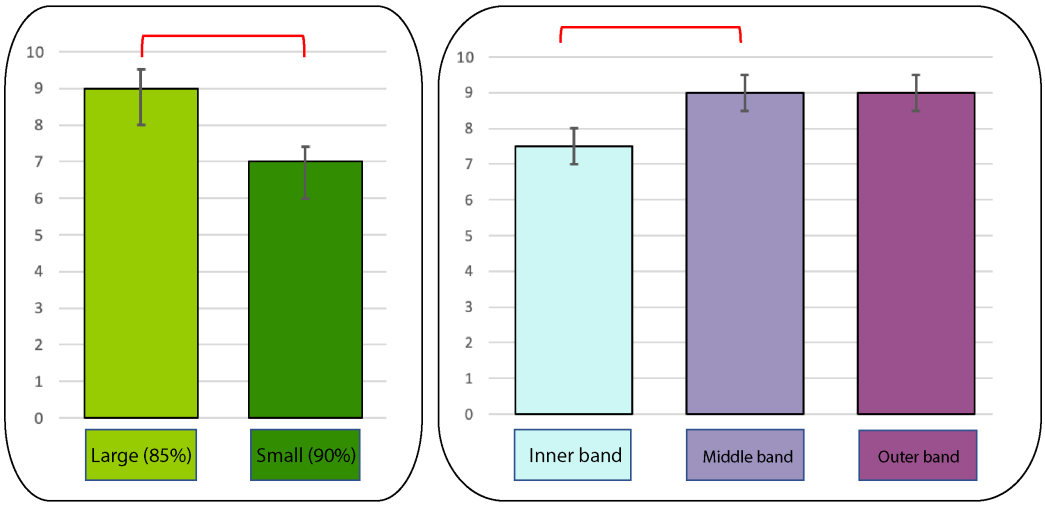


Figure 6.11: Median of failed trials split by size of the interaction area (left) and three bands EdgeSelect (right)

6.3.5.4 *Qualitative Analysis*

At the end of the experiment, we asked all of our participants to fill out a questionnaire about the interaction technique and experiment. We asked them to rank the interaction layers (Inner-, Middle-, and Outer-band) based on their preference and easiness. In addition, we asked our participants which interaction area size they preferred (Large or Small). Seventy-five percent of our participants chose the Large area. Many of our participants explained that it was easier to interact with compared to the Small interaction area. For instance, P6 commented, "Touch area is large and easy to touch", P5 and P3 also mentioned that they did not have any "accidental touch in the large area" and that it "feels more precise" in the large interaction area. In addition, nearly 60% of the participants preferred to use the Outer-band to interact with graphs (Figure 6.12). Only 8% of the participants picked the Inner-band as their favourite layer to interact with.

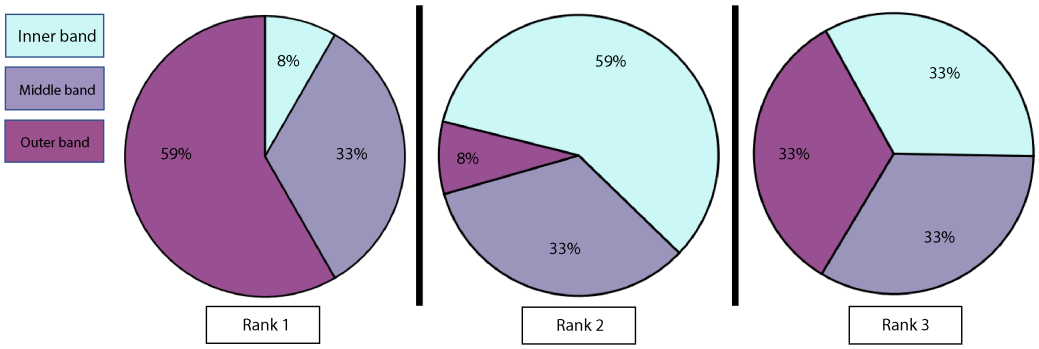


Figure 6.12: Percentage of each of the three bands' rankings. Outer-band was the most preferred layer of EdgeSelect. Nearly 60% of participants preferred to interact with this Outer-layer (left).

6.3.6 *Discussion*

Our results show that the Small interaction area enables better target selection performance than the Large interaction area in terms of response times and the maximum graph density that participants could interact with. Our data analysis showed that interaction with the Small area yields a significantly faster performance (almost 500ms faster) than the Large area. When the interaction area is smaller, it is faster to scroll and navigate through the interaction area, which means that the user can interact with more data points in a shorter period of time.

The average maximum number of targets our participants could reach, while interacting with the Small area was slightly, but significantly, higher ithen with the Large interaction area. The smaller size made the interaction faster to reach the targets, and that means even if participants made a mistake in target selection, they had time to correct their selection. In the Large interaction area, since it took longer for our participants to navigate through the interaction area, correcting a target selection error could take longer, which could be considered as a fail trial.

In addition, the minimum number of targets participants reached was significantly higher in the Small condition, which means they made fewer mistakes while interacting with the Small interaction area. Figure 6.9-left also shows that the number of failures was lower in the Small interaction area.

Although there was no difference between the three different layers of the EdgeSelect interaction area across the Small and Large trials, in terms of response time (Figure 6.8-right), the number of targets they could reach using the Middle-band was higher compared to other bands. This means that the Middle-band is a good option to interact with more dense graphs with a higher number of data points.

Although our quantitative data analysis shows that the small interaction area was better than the large interaction area, in terms of performance, our post-experiment interview shows that they preferred the large interaction area over the small interaction area. One hypothesis is that most of our participants used their fingertips in the interaction area and placed their fingers perpendicular to the screen in the small interaction area condition. This could result in more control over the interaction and performing the task. Whereas in the large interaction area, participants used their fingers, tilted, and utilized a larger fingertip area to interact with the smartwatch display, a more natural way of interacting with smartphones and smartwatch displays. We think that interacting with the small interaction area was less natural, which could be why they did not prefer it over the large interaction area.

6.4 GENERAL DISCUSSION

In this section we discuss the critical findings and potential applications, and limitations of the EdgeSelect technique.

6.4.1 *Key Findings*

Result of the first study reveal that interacting with the segments at the bottom, bottom-right, and right side of the smartwatch display yields the best screen visibility, with an average of 86.7%, 90% and 85% screen visibility, compared to other segments of the smartwatch display. As we move toward the center of the smartwatch display and the top-left corner, the screen visibility decreases significantly. Interaction with the top-left corner of the smartwatch display has the worst screen visibility with an average of only 40.9% of the screen being visible. The outermost segments at the right-bottom corner of the display offer a 92% or higher screen visibility, confirming the result reported by [73], which measured the screen occlusion only for the segments of the smartwatch bezel. However, our experiment showed that interacting with inner segments of the smartwatch display can also provide a reasonable amount of visibility to users (more than 80% screen visibility). Including these specific areas of the smartwatch display, as the interaction area, without sacrificing the screen visibility, can help smartwatch app designers to use the width and length of EdgeSelect to select and interact with graphs or other types of content that requires target selection and continuous linear interaction.

In the second experiment, we showed that using segments of the smartwatch display with at least 90% screen visibility (Small interaction area) outperformed the interaction with the segments of the smartwatch screen with at least 85% screen visibility (Large interaction area), in terms of response time and errors. This means that the Small interaction area occupies less smartwatch screen real estate and enables faster and more accurate user interaction with graphs and data points. We also found that the Middle-band in the interaction area is more suitable to interact with graphs with a higher number of data points (e.g., line/bar graphs) compared to Inner- and Outer-band which are suitable to interact with less dense graphs (e.g., pie/donut chart).

6.4.2 *Potential Applications*

EdgeSelect can be used to interact with content other than graphs such as virtual keyboards (Figure 6.13-a), sliders (smartwatch setting, Figure 6.13-b), navigating a music track (Figure 6.13-c), and hierarchical menu selection (Figure 6.13-d). EdgeSelect can be used to interact with three-row virtual keyboards. Each row of the virtual keyboard can be mapped to each band of the interaction area. To make the interaction intuitive, the top-row of the virtual keyboard can be mapped to the inner-band, middle-row to middle-band and the bottom-row of the keyboard to the outer-band of the EdgeSelect. There are approximately nine letters per row (Figure 6.13-a). As the results of our second experiment suggest, using the EdgeSelect can effectively interact with such a number of points in each row of the virtual keyboard. By sliding the index finger on the right band, the smartwatch user can select the right letter they want to type. Selection can be made by moving

the finger a bit higher than the smartwatch display or by holding the finger in that position for a short period of time (e.g., 2 seconds).

Interacting with sliders (e.g., increasing and decreasing smartwatch brightness and sound volume) and similarly navigating through a music player are two other examples of continuous interaction using EdgeSelect. EdgeSelect can also be used to interact with nested or hierarchical menus. The first layer of the interaction area can interact with the highest level of the menu. Then there will be items from the sub-menu (second row, Figure 6.13-d). The second band can be used to select the item from the sub-menu. The same process may happen with the third layer of the menu and the last band of the interaction area. For instance, the user can select a workout application from the apps menu, then the type of workout (e.g., running) and then the duration s/he wants to workout.

Figure 6.13: a) Each of the three EdgeSelect bands can be used to interact with a three-row virtual keyboard on the smartwatch without blocking the letter while typing. b, c) Using EdgeSelect to interact with sliders such as adjusting settings and navigating through music tracks. d) EdgeSelect can also be used in hierarchical menus for item selection.

6.4.2.1 *Limitations and Future Work*

We decided to use arrays of targets in our target selection experiment as using actual graphs would add more complexity by adding too many confounding

variables. Although we demonstrated that EdgeSelect was designed to interact with graphs with different densities (Figure 6.6), formal qualitative and quantitative evaluation could confirm the effectiveness of EdgeSelect on actual graphs. In the future, we will quantify the efficiency of using EdgeSelect to interact with various types of graphs with different levels of density.

Another limitation of this work is that the two experiments were done in the lab. Smartwatch users employ their smartwatches on-the-go and in different mobility conditions. Running an in-the-wild study with participants running and being outdoors, would make data collection (such as measuring the screen visibility) very difficult and inaccurate. many factors, such as lighting, could affect the result of our video processing. To reduce these types of confounding factors, we excluded mobility conditions from our current studies and controlled the environment by conducting a lab based-study. However, in reality, environment factors are constantly changing. We will further explore EdgeSelect under different ecologically valid settings.

6.5 CONCLUSION

In this chapter we introduce EdgeSelect, an interaction technique designed to facilitate exploring multiple data visualizations on small smartwatch screens. EdgeSelect was inspired by the need to interact with various graphs, representing multiple interrelated data sources, which can often be shown at the same time on a smartwatch. Our design was geared at mitigating 'fat finger' and screen occlusion effects, as the interacting finger largely overlaps the small display when examining content. One of our key design goals was to shift the interactive region to an area

of the smartwatch resulting in minimal screen occlusion (about 10% in our study) while optimizing input across the entire smartwatch display. To design EdgeSelect, we first conducted a study to measure the screen occlusion caused by the finger interacting with the entire smartwatch display. Our results indicate that interacting with outer segments of the smartwatch display to the right, bottom-right and bottom sides of the display offers the best screen visibility. Based on this initial result, we designed a three-layer EdgeSelect interaction technique. Each of the three layers can be mapped to interact with different graphs, making EdgeSelect one of the first interaction techniques enabling the exploration of multiple data visualizations on a small smartwatch display.

In the second experiment, we examined EdgeSelect with two different sizes of the interaction area, Large and Small, which enable inversely proportional levels of screen visibility. Our results showed that participants performed faster and more accurately with the Small interaction area (with higher screen visibility) compared to the Large interaction area (with lower screen visibility), making EdgeSelect a suitable interaction method on the small smartwatch display. We also found that the Middle-band of EdgeSelect is more suitable to interact with denser graphs (e.g., line/bar chart). On the other hand, Outer- and Inner-bands are more suitable to interact with less dense graphs (pie/donut charts). Finally, we demonstrated that EdgeSelect enables interaction with multiple graphs of different types and densities, and also can be applied to interact with a wide range of other applications, including text-entry, menu selection, and adjusting sliders. In future work, we aim to deploy EdgeSelect in ecologically valid settings to examine the breadth of exploratory in-situ analysis afforded by such a technique.

7 CONCLUSION

We believe that smartwatches have specific features that make them unique compared to other devices. One of the main characteristics of a smartwatch is its small form factor. However, the small size of the smartwatch display can be considered as one of the limitations of such devices to visualize and interact with the collected data. Also, It has been reported by previous research that existing visualization techniques on smartwatch applications are not suitable and can be improved significantly. Interacting with visualization techniques is another topic of this thesis. We showed how novel interaction techniques with visualizations can help the smartwatch user to explore the collected data. This thesis undertakes a broad investigation of data visualization and interaction with visualizations on the small display of smartwatches.

Chapter 1 explained the major objectives of this thesis and also the motivation behind these goals. We discussed how the small size of the smartwatch display could be considered as one of the most significant limitations of smartwatches to present the collected data. New visualization techniques should be developed to overcome this limitation. Therefore, we proposed two approaches, "Compression" and "Simplification," to optimize the available space. We discussed these two techniques in chapter 3 and chapter 4, respectively. Previous research addressed the lack of proper interaction methods with visualizations as one of the main

reasons preventing the smartwatch users from further data exploration. In chapters 5 and 6, we discussed two different approaches to interact with graphs and charts on the small display of smartwatches.

Chapter 2 discussed the related work, background and recent progress related to data representation and interaction with visualizations on smartwatches. The main focus of this section is to show the significant issues to visualize and interact with the data on the small display of smartwatches. We also reported the recent related work and techniques to overcome existing limitations on smartwatches, suggested by previous research. We then showed the existing gaps in the literature and potential ways to fix them.

Chapter 3 discussed the Compression technique as one of the ways to optimize the space on the small display of smartwatches. In this chapter, we evaluated three different compression approaches (along the x-axis, y-axis and xy-axes). By conducting an experiment, we realized that condensing a line graph on the x-axis could improve the users' performance performing visual tasks. Therefore, we introduced G-Spark, which shrinks a line graph on the x-axis to the point of representing one data sample per pixel. We showed that G-Spark successfully minimized the interaction between the user and the chart without sacrificing the glanceability on smartwatches. In addition, the results of our experiment showed us some improvements in users' performance and accuracy. Therefore, we defined G-Spark as a compact visual representation of glanceable line graphs for smartwatches.

Chapter 4 discussed the simplification technique as a tool to generate more available space on data visualization. These generated available spaces could be beneficial to add auxiliary information besides the main visualization. This plays a significant role in the small display of smartwatches where there is a

limited space to represent additional interlinked datasets. With two experiments, we showed that our simplification algorithm, SF-LG, could generate more space around the primary line graph. To evaluate our algorithm, we compared it with PIP, which is a well-known simplification method. Our results showed that SF-LG was significantly faster compared to PIP and also non-simplified graphs.

Chapter 5 looked into interaction with data visualization on smartwatches as a tool for further data exploration. However, the limited size of the smartwatch display is a barrier for direct interaction with the screen. The 'fat finger' and screen occlusion are two of the most common issues while interacting with such small displays. To overcome these challenges, we proposed BezelGlide, a new interaction technique specifically designed to interact with a wide range of visualizations that uses smartwatch bezels to interact with the content on the display. Our experiment showed that this technique minimized the screen occlusion and solved the 'fat finger' problem. We also evaluated BezelGlid with the Shift techniques, which is a well-known interaction method to overcome the fat finger problem. The results showed that BezelGlid was significantly faster compared to the Shift technique.

In chapter 6 , we examined the possibility of minimizing the interactive region of a smartwatch display, to mitigate screen occlusion while exploring multiple graphs. This small interactive region, is designed to occupy a small percent of the smartwatch display without hindering the density of graphs displayed on the watch.

7.1 LIMITATIONS

For all the experiments conducted in this thesis there were limitations explained in each dedicated chapter. However, in this section we will discuss the overall limitations of this thesis.

Young participants. The average age of our participants in all experiments is less than 30. For some of the experiments (two studies of chapter 6) the average age of participants is approximately 20 years. As most of our studies were conducted in a lab environment at the University of Manitoba, most of our participants were university students. This could be important because as the age goes higher, participants' performance and reaction time to perform the tasks could be slower than young participants in general. This means that ideally, having participants from various ranges of ages could be beneficial to generalize the result of our experiment to a more significant population. Also, although we had female participants in all of our experiments, we had a dominant number of male participants in our experiments. This was due to having fewer female volunteers in the studies.

The other limitation of our studies is that we mainly focused on line graphs because a line graph is a space-efficient visualization technique common on smartwatches, and it is the most common way to visualize time series data. However, there are many other different types of graphs on smartwatches. Although we demonstrated potential applications and generalizability of our techniques to other graph types in different sections, we did not implement and evaluate our methods on other types of charts. For instance, in chapter 5, Partial-BezelGlide was made and considered for line graphs; however, it is not clear if it can be an

effective technique on bar and donut charts which are two of the most common visualization techniques on smartwatches. In addition, line graphs were used in the SF-LG technique; however, it is unclear if such a space-efficient approach can be generalized to other graphs similar to line graphs. For instance, we do not know whether a bar chart, which is not a space-efficient visualization technique, can be more space-efficient after applying SF-LG or not. Also, in the first chapter, the effect of compression was tested on line graphs, but the generalizability of GSpark to other charts was not evaluated.

We conducted all the experiments in a lab environment, mainly to have more control over variables and prevent adding more complexity to the experiments. Although we designed our experiments to be as natural as possible, there were differences between a realistic condition and our experimental setups. This can be considered one of our experiments' limitations as our experiments differed from natural smartwatch usage conditions. For instance, in BezelGlid second experiment, we asked participants to walk on a treadmill at a constant speed (their preferred walking speed), constantly interacting with the smartwatch. However, in a realistic situation, many smartwatch users use their smartwatch to walk/run on the field (not treadmill). Walking on a field requires more attention, which means more cognitive load, affecting the users' performance understanding/interacting with graphs.

7.2 FUTURE WORK

In this section, we will discuss the potential general ideas that can be considered as future work to improve data visualization techniques on smartwatches.

One of the main potential ideas which can be studies in the future is to investigate how we can improve various other graph types on small screen of smartwatches. As it was mentioned in the the previous section, our studies was focused on the line graphs. However, our data visualization techniques, including compression and simplification techniques, can be generalized to other graph types that represent complex times series data (e.g., bar chart). Although, in this thesis, we improved the existing line graphs on smartwatches using both simplification and compression techniques, novel new visualization techniques could be designed to visualize various types of data. These visualization techniques can be proposed based on factors such as type of data, shape of the screen (circular or rectangular smartwatch display), and different queries smartwatch users have from the graphs (visual exploration task).

When it comes to data visualization techniques on smartwatches, there is no clear definition of glanceable data visualization. Although there are numbers reported by previous researchers [12], those studies were conducted in the controlled lab environment. They were focused on particular visualization techniques (bar, donut, and radial bar graph) with a very specific task (data points comparison task). For instance, Blascheck et al. [12] reported that participants performed the data points comparison task in less than 4 seconds which was significantly higher compared with the bar and donut chart. However, we think 3.5 seconds is still a reasonable time to read and understand a complex graph and perform a task. The glanceability of a graph depends on many factors, including the type of the graph, the number of data points, the primary task that the user performs (e.g., working out), and visual characteristics of graphs (e.g., size and colour). A potential future work could be about how different combinations of these factors affect a graph's glanceability on a small smartwatch screen. Another idea could

be about defining the proper definition of glanceable visualization. In this regard, many challenges should be considered. For instance, a visual exploration task could be very complex by its nature (e.g., the exact difference between two data points), which means it will take a relatively long time for smartwatch users to perform the task. If we assume that a smartwatch user can perform this task as fast as possible, but in 60 seconds, do we call this visualization/task a glanceable visualization because the response time was the fastest it could be?

Adding different mobility conditions to smartwatch studies is necessary as smartwatch users use their smartwatch in various mobility conditions (e.g., sitting, standing, walking and running). For instance, data visualization techniques can help runners adjust their speed to achieve their goals. However, it is not clear if running such experiments in a lab environment is similar to conducting such experiments in more realistic conditions. For instance, a study can be done on a treadmill with a fixed speed; however, the same study can be done on an open field, giving the participants more flexibility. Hence, as future work, we can run an experiment with two conditions, the in-lab condition versus on the field condition to see the differences between the two conditions.

BIBLIOGRAPHY

[1] Muhammad Adnan, Mike Just, and Lynne Baillie. "Investigating time series visualisations to improve the user experience." In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2016, pp. 5444–5455.

[2] Sunggeun Ahn, Jaeyeon Lee, Keunwoo Park, and Geehyuk Lee. "Evaluation of edge-based interaction on a square smartwatch." In: *International Journal of Human-Computer Studies* 109 (2018), pp. 68–78.

[3] Ozgur Akyazi, Sahin Batmaz, Bilgin Kosucu, and Bert Arnrich. "SmokeWatch: A smartwatch smoking cessation assistant." In: *2017 25th Signal Processing and Communications Applications Conference (SIU)*. IEEE. 2017, pp. 1–4.

[4] Reem Albaghli and Kenneth M Anderson. "A vision for heart rate health through wearables." In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. 2016, pp. 1101–1105.

[5] Fereshteh Amini, Khalad Hasan, Andrea Bunt, and Pourang Irani. "Data representations for in-situ exploration of health and fitness data." In: *Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare*. 2017, pp. 163–172.

[6] Amir Mohammad Amiri, Nicholas Peltier, Cody Goldberg, Yan Sun, Anoo Nathan, Shivayogi V Hiremath, and Kunal Mankodiya. "WearSense: de-

tecting autism stereotypic behaviors through smartwatches." In: *Healthcare*. Vol. 5. 1. Multidisciplinary Digital Publishing Institute. 2017, p. 11.

[7] Shaikh Shawon Arefin Shimon, Courtney Lutton, Zichun Xu, Sarah Morrison-Smith, Christina Boucher, and Jaime Ruiz. "Exploring non-touchscreen gestures for smartwatches." In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2016, pp. 3822–3833.

[8] Karl Johan Åström. "On the choice of sampling rates in parametric identification of time series." In: *Information Sciences* 1.3 (1969), pp. 273–278.

[9] Sandra Bardot, Marcos Serrano, and Christophe Jouffrais. "From tactile to virtual: using a smartwatch to improve spatial map exploration for visually impaired users." In: *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 2016, pp. 100–111.

[10] Vivien Beattie and Michael John Jones. "The impact of graph slope on rate of change judgments in corporate reports." In: *Abacus* 38.2 (2002), pp. 177–199.

[11] Chelsea G Bender, Jason C Hoffstot, Brian T Combs, Sara Hooshangi, and Justin Cappos. "Measuring the fitness of fitness trackers." In: *2017 IEEE Sensors Applications Symposium (SAS)*. IEEE. 2017, pp. 1–6.

[12] Tanja Blascheck, Lonni Besançon, Anastasia Bezerianos, Bongshin Lee, and Petra Isenberg. "Glanceable visualization: Studies of data comparison performance on smartwatches." In: *IEEE transactions on visualization and computer graphics* 25.1 (2018), pp. 630–640.

[13] Costas Boletsis, Simon McCallum, and Brynjar Fowels Landmark. "The use of smartwatches for health monitoring in home-based dementia care."

In: *International Conference on Human Aspects of IT for the Aged Population*. Springer. 2015, pp. 15–26.

[14] Danielle Bragg, Shiri Azenkot, and Adam Tauman Kalai. "Reading and Learning Smartfonts." In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 2016, pp. 391–402.

[15] Emeline Brulé, Gilles Bailly, Marcos Serrano, Marc Teyssier, and Samuel Huron. "Investigating the design space of smartwatches combining physical rotary inputs." In: *Proceedings of the 29th Conference on l'Interaction Homme-Machine*. 2017, pp. 13–20.

[16] Maria Beatriz Carmo, Ana Paula Afonso, and Paulo Pombinho Matos. "Visualization of geographic query results for small screen devices." In: *Proceedings of the 4th ACM workshop on Geographical information retrieval*. 2007, pp. 63–64.

[17] Jaemin Chun, Anind Dey, Kyungtaek Lee, and SeungJun Kim. "A qualitative study of smartwatch usage and its usability." In: *Human Factors and Ergonomics in Manufacturing & Service Industries* 28.4 (2018), pp. 186–199.

[18] Fu-Lai Chung, Tak-Chung Fu, Robert Luk, Vincent Ng, et al. "Flexible time series pattern matching based on perceptually important points." In: (2001).

[19] William S Cleveland and Robert McGill. "Graphical perception: Theory, experimentation, and application to the development of graphical methods." In: *Journal of the American statistical association* 79.387 (1984), pp. 531–554.

[20] William S Cleveland and Robert McGill. "An experiment in graphical perception." In: *International Journal of Man-Machine Studies* 25.5 (1986), pp. 491–500.

[21] Jacob Cohen. *Statistical power analysis for the behavioral sciences*. Academic press, 2013.

[22] Sunny Consolvo, Beverly Harrison, Ian Smith, Mike Y Chen, Katherine Everitt, Jon Froehlich, and James A Landay. "Conducting in situ evaluations for and with ubiquitous computing technologies." In: *International Journal of Human-Computer Interaction* 22.1-2 (2007), pp. 103–118.

[23] Frederick E Croxton and Harold Stein. "Graphic comparisons by bars, squares, circles, and cubes." In: *Journal of the American Statistical Association* 27.177 (1932), pp. 54–60.

[24] Frederick E Croxton and Roy E Stryker. "Bar charts versus circle diagrams." In: *Journal of the American Statistical Association* 22.160 (1927), pp. 473–482.

[25] Oscar De Bruijn, Robert Spence, and Min Yih Chong. "RSVP browser: Web browsing on small screen devices." In: *Personal and Ubiquitous Computing* 6.4 (2002), pp. 245–252.

[26] Walter Crosby Eells. "The relative merits of circles and bars for representing component parts." In: *Journal of the American Statistical Association* 21.154 (1926), pp. 119–132.

[27] Frederic Ehrler and Christian Lovis. "Supporting elderly homecare with smartwatches: advantages and drawbacks." In: *Studies in health technology and informatics* 205 (2014), pp. 667–71.

[28] Andrey Esakia, D Scott McCrickard, Samantha Harden, and Michael Horning. "FitAware: Promoting Group Fitness Awareness ThroughGlanceable Smartwatches." In: *Proceedings of the 2018 ACM Conference on Supporting Groupwork*. 2018, pp. 178–183.

[29] Paolo Federico, Stephan Hoffmann, Alexander Rind, Wolfgang Aigner, and Silvia Miksch. "Qualizon graphs: Space-efficient time-series visualization with qualitative abstractions." In: *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*. 2014, pp. 273–280.

[30] Tak-chung Fu, Fu-lai Chung, Ka-yan Kwok, and Chak-man Ng. "Stock time series visualization based on data point importance." In: *Engineering Applications of Artificial Intelligence* 21.8 (2008), pp. 1217–1232.

[31] Tak chung Fu, Fu lai Chung, Robert Luk, and Chak man Ng. "Stock time series pattern matching: Template-based vs. rule-based approaches." In: *Engineering Applications of Artificial Intelligence* 20.3 (2007), pp. 347 –364. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2006.07.003. URL: http://www.sciencedirect.com/science/article/pii/S0952197606001278.

[32] Johannes Fuchs, Fabian Fischer, Florian Mansmann, Enrico Bertini, and Petra Isenberg. "Evaluation of alternative glyph designs for time series data in a small multiple setting." In: *Proceedings of the SIGCHI conference on human factors in computing systems*. 2013, pp. 3237–3246.

[33] Michael Fulk. "Improving web browsing on handheld devices." In: *CHI'01 extended abstracts on Human factors in computing systems*. 2001, pp. 395–396.

[34] Rúben Gouveia, Fábio Pereira, Evangelos Karapanos, Sean A Munson, and Marc Hassenzahl. "Exploring the design space of glanceable feedback for physical activity trackers." In: *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 2016, pp. 144–155.

[35] Wei Guo and Jingtao Wang. "SmartRSVP: Facilitating Attentive Speed Reading on Small Screen Wearable Devices." In: *Proceedings of the 2017 CHI*

Conference Extended Abstracts on Human Factors in Computing Systems. 2017, pp. 1640–1647.

[36] Xiaonan Guo, Jian Liu, and Yingying Chen. "FitCoach: Virtual fitness coach empowered by wearable mobile devices." In: *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE. 2017, pp. 1–9.

[37] Sean G Gustafson and Pourang P Irani. "Comparing visualizations for tracking off-screen moving targets." In: *CHI'07 Extended Abstracts on Human Factors in Computing Systems*. 2007, pp. 2399–2404.

[38] Sean Gustafson, Patrick Baudisch, Carl Gutwin, and Pourang Irani. "Wedge: clutter-free visualization of off-screen locations." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2008, pp. 787–796.

[39] Radim Halır and Jan Flusser. "Numerically stable direct least squares fitting of ellipses." In: *Proc. 6th International Conference in Central Europe on Computer Graphics and Visualization. WSCG*. Vol. 98. Citeseer. 1998, pp. 125–132.

[40] Teng Han, Khalad Hasan, Keisuke Nakamura, Randy Gomez, and Pourang Irani. "Soundcraft: Enabling spatial interactions on smartwatches using hand generated acoustics." In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 2017, pp. 579–591.

[41] Teng Han, Jiannan Li, Khalad Hasan, Keisuke Nakamura, Randy Gomez, Ravin Balakrishnan, and Pourang Irani. "PageFlip: Leveraging Page-Flipping Gestures for Efficient Command and Value Selection on Smartwatches." In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, pp. 1–12.

[42] Ming C Hao, Manish Marwah, Halldór Janetzko, Umeshwar Dayal, Daniel A Keim, Debprakash Patnaik, Naren Ramakrishnan, and Ratnesh K Sharma. "Visual exploration of frequent patterns in multivariate time series." In: *Information Visualization* 11.1 (2012), pp. 71–83.

[43] Kiyotaka Hara, Takeshi Umezawa, and Noritaka Osawa. "Effect of button size and location when pointing with index finger on smartwatch." In: *International Conference on Human-Computer Interaction*. Springer. 2015, pp. 165–174.

[44] Khalad Hasan, Xing-Dong Yang, Hai-Ning Liang, and Pourang Irani. "How to position the cursor? an exploration of absolute and relative cursor positioning for back-of-device input." In: *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*. 2012, pp. 103–112.

[45] Jeffrey Heer and Michael Bostock. "Crowdsourcing graphical perception: using mechanical turk to assess visualization design." In: *Proceedings of the SIGCHI conference on human factors in computing systems*. 2010, pp. 203–212.

[46] Alfredo I Hernandez, Fernando Mora, M Villegas, Gianfranco Passariello, and Guy Carrault. "Real-time ECG transmission via Internet for nonclinical applications." In: *IEEE Transactions on information technology in biomedicine* 5.3 (2001), pp. 253–257.

[47] Dandan Huang, Melanie Tory, Bon Adriel Aseniero, Lyn Bartram, Scott Bateman, Sheelagh Carpendale, Anthony Tang, and Robert Woodbury. "Personal visualization and personal visual analytics." In: *IEEE Transactions on Visualization and Computer Graphics* 21.3 (2014), pp. 420–433.

[48] Stéphane Huot and Eric Lecolinet. "Focus+ context visualization techniques for displaying large lists with multiple points of interest on small tactile screens." In: *IFIP Conference on Human-Computer Interaction*. Springer. 2007, pp. 219–233.

[49] Alaul Islam, Anastasia Bezerianos, Bongshin Lee, Tanja Blascheck, and Petra Isenberg. "Visualizing information on watch faces: A survey with smartwatch users." In: *arXiv preprint arXiv:2009.00750* (2020).

[50] Waqas Javed, Bryan McDonnel, and Niklas Elmqvist. "Graphical perception of multiple time series." In: *IEEE transactions on visualization and computer graphics* 16.6 (2010), pp. 927–934.

[51] Brian Johnson and Ben Shneiderman. "Tree-maps: A space-filling approach to the visualization of hierarchical information structures." In: *Readings in Information Visualization: Using Vision to Think* (1999), pp. 152–159.

[52] Emil Jovanov. "Preliminary analysis of the use of smartwatches for longitudinal health monitoring." In: *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE. 2015, pp. 865–868.

[53] Amy K Karlson and Benjamin B Bederson. "ThumbSpace: generalized one-handed input for touchscreen-based mobile devices." In: *IFIP Conference on Human-Computer Interaction*. Springer. 2007, pp. 324–338.

[54] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. "Dimensionality reduction for fast similarity search in large time series databases." In: *Knowledge and information Systems* 3.3 (2001), pp. 263–286.

[55] Eamonn Keogh, Harry Hochheiser, and Ben Shneiderman. "An augmented visual query mechanism for finding patterns in time series data." In: *International Conference on Flexible Query Answering Systems*. Springer. 2002, pp. 240–250.

[56] Frederic Kerber, Tobias Kiefer, Markus Löchtefeld, and Antonio Krüger. "Investigating current techniques for opposite-hand smartwatch interaction." In: *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 2017, pp. 1–12.

[57] Jayden Khakurel, Antti Knutas, Helinä Melkas, Birgit Penzenstadler, Bo Fu, and Jari Porras. "Categorization framework for usability issues of smartwatches and pedometers for the older adults." In: *International Conference on Universal Access in Human-Computer Interaction*. Springer. 2018, pp. 91–106.

[58] Nicholas Kong, Jeffrey Heer, and Maneesh Agrawala. "Perceptual guidelines for creating rectangular treemaps." In: *IEEE transactions on visualization and computer graphics* 16.6 (2010), pp. 990–998.

[59] Yuki Kubo, Buntarou Shizuki, and Jiro Tanaka. "B2B-Swipe: Swipe gesture for rectangular smartwatches from a bezel to a bezel." In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2016, pp. 3852–3856.

[60] Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. "Task taxonomy for graph visualization." In: *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*. 2006, pp. 1–5.

[61] Luis A Leiva, Alireza Sahami, Alejandro Catala, Niels Henze, and Albrecht Schmidt. "Text entry on tiny qwerty soft keyboards." In: *Proceedings of the*

33rd Annual ACM Conference on Human Factors in Computing Systems. 2015, pp. 669–678.

[62] Ian Li, Anind K Dey, and Jodi Forlizzi. "Understanding my data, myself: supporting self-reflection with ubicomp technologies." In: *Proceedings of the 13th international conference on Ubiquitous computing*. 2011, pp. 405–414.

[63] Jessica Lin, Eamonn Keogh, and Stefano Lonardi. "Visualizing and discovering non-trivial patterns in large time series databases." In: *Information visualization* 4.2 (2005), pp. 61–82.

[64] Mark A Livingston, Jonathan W Decker, and Zhuming Ai. "Evaluation of multivariate visualization on a multivariate task." In: *IEEE transactions on visualization and computer graphics* 18.12 (2012), pp. 2114–2121.

[65] Kent Lyons. "What can a dumb watch teach a smartwatch? Informing the design of smartwatches." In: *Proceedings of the 2015 ACM international symposium on wearable computers*. 2015, pp. 3–10.

[66] Kent Lyons, David Nguyen, Daniel Ashbrook, and Sean White. "Facet: a multi-segment wrist worn system." In: *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 2012, pp. 123–130.

[67] Meethu Malu and Leah Findlater. "Toward accessible health and fitness tracking for people with mobility impairments." In: *Proceedings of the 10th EAI International Conference on Pervasive Computing Technologies for Healthcare*. 2016, pp. 170–177.

[68] Donald McMillan, Barry Brown, Airi Lampinen, Moira McGregor, Eve Hoggan, and Stefania Pizza. "Situating wearables: Smartwatch use in context."

In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2017, pp. 3582–3594.

[69] Jochen Meyer, Anastasia Kazakova, Merlin Büsing, and Susanne Boll. "Visualization of complex health data on mobile devices." In: *Proceedings of the 2016 ACM Workshop on Multimedia for Personal Health and Health Care*. 2016, pp. 31–34.

[70] Babak Moatamed, Farhad Shahmohammadi, Ramin Ramezani, Arash Naeim, Majid Sarrafzadeh, et al. "Low-cost indoor health monitoring system." In: *2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*. IEEE. 2016, pp. 159–164.

[71] Subhas Chandra Mukhopadhyay. "Wearable sensors for human activity monitoring: A review." In: *IEEE sensors journal* 15.3 (2014), pp. 1321–1330.

[72] Ali Neshati, Fouad Alallah, Bradley Rey, Yumiko Sakamoto, Marcos Serrano, and Pourang Irani. "SF-LG: Space-Filling Line Graphs for Visualizing Interrelated Time-series Data on Smartwatches." In: *Proceedings of the 23rd International Conference on Mobile Human-Computer Interaction*. 2021, pp. 1–13.

[73] Ali Neshati, Bradley Rey, Ahmed Shariff Mohommed Faleel, Sandra Bardot, Celine Latulipe, and Pourang Irani. "BezelGlide: Interacting with Graphs on Smartwatches with Minimal Screen Occlusion." In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 2021, pp. 1–13.

[74] Ali Neshati, Yumiko Sakamoto, Launa C Leboe-McGowan, Jason Leboe-McGowan, Marcos Serrano, and Pourang Irani. "G-Sparks: Glanceable Sparklines on Smartwatches." In: *Graphics Interface*. 2019, pp. 23–1.

[75] Alexander Ng, John Williamson, and Stephen Brewster. "The effects of encumbrance and mobility on touch-based gesture interactions for mobile phones." In: *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 2015, pp. 536–546.

[76] Anne Ngu, Yeahuay Wu, Habil Zare, Andrew Polican, Brock Yarbrough, and Lina Yao. "Fall detection using smartwatch sensor data with accessor architecture." In: *International Conference on Smart Health*. Springer. 2017, pp. 81–93.

[77] Ian Oakley and Doyoung Lee. "Interaction on the edge: offset sensing for small devices." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2014, pp. 169–178.

[78] Ian Oakley, Carina Lindahl, Khanh Le, DoYoung Lee, and MD Rasel Islam. "The flat finger: Exploring area touches on smartwatches." In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2016, pp. 4238–4249.

[79] Julie Pallant. *SPSS survaival manual: a step by step guide to data analysis using SPSS*. 2010.

[80] Keunwoo Park, Daehwa Kim, Seongkook Heo, and Geehyuk Lee. "MagTouch: Robust Finger Identification for a Smartwatch Using a Magnet Ring and a Built-in Magnetometer." In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI '20. Honolulu, HI, USA: Association for Computing Machinery, 2020, 1–13. ISBN: 9781450367080. DOI: 10.1145/3313831.3376234. URL: https://doi.org/10.1145/3313831.3376234.

[81] J Parnow. "Micro visualizations: How can micro visualisations enhance text comprehension, memorability, and exploitation?" In: *Design MS Thesis, Potsdam Univ. Appl. Sci., Potsdam, Germany* (2015).

[82] Soubhik Paul, Jayanta Mukhopadhyay, Arun K Majumdar, Bandana Majumdar, and S Das Bhattacharya. "Methodology to visualize electronic health record for chronic diseases on small display screens." In: *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*. 2012, pp. 505–510.

[83] Charles Perin, Frédéric Vernier, and Jean-Daniel Fekete. "Interactive horizon graphs: Improving the compact visualization of multiple time series." In: *Proceedings of the SIGCHI conference on human factors in computing systems*. 2013, pp. 3217–3226.

[84] Lewis V Peterson and Wilbur Schramm. "How accurately are different kinds of graphs read?" In: *Audiovisual communication review* 2.3 (1954), pp. 178–189.

[85] Dung Phan, Lee Yee Siong, Pubudu N Pathirana, and Aruna Seneviratne. "Smartwatch: Performance evaluation for long-term heart rate monitoring." In: *2015 International symposium on bioelectronics and bioinformatics (ISBB)*. IEEE. 2015, pp. 144–147.

[86] Stefania Pizza, Barry Brown, Donald McMillan, and Airi Lampinen. "Smartwatch in vivo." In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2016, pp. 5456–5469.

[87] Katrin Plaumann, Michael Müller, and Enrico Rukzio. "CircularSelection: optimizing list selection for smartwatches." In: *Proceedings of the 2016 ACM International Symposium on Wearable Computers*. 2016, pp. 128–135.

[88] Blaine Reeder and Alexandria David. "Health at hand: a systematic review of smart watch uses for health and wellness." In: *Journal of biomedical informatics* 63 (2016), pp. 269–276.

[89] George Robertson, Roland Fernandez, Danyel Fisher, Bongshin Lee, and John Stasko. "Effectiveness of animation in trend visualization." In: *IEEE transactions on visualization and computer graphics* 14.6 (2008), pp. 1325–1332.

[90] Paul Rosen and Ghulam Jilani Quadri. "LineSmooth: An Analytical Framework for Evaluating the Effectiveness of Smoothing Techniques on Line Charts." In: *IEEE Transactions on Visualization and Computer Graphics* (2020).

[91] Volker Roth and Thea Turner. "Bezel swipe: conflict-free scrolling and multiple selection on mobile touch screen devices." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2009, pp. 1523–1526.

[92] Virpi Roto, Andrei Popescu, Antti Koivisto, and Elina Vartiainen. "Minimap: a web page visualization method for mobile phones." In: *Proceedings of the SIGCHI conference on Human Factors in computing systems*. 2006, pp. 35–44.

[93] Anne Roudaut, Stéphane Huot, and Eric Lecolinet. "TapTap and MagStick: improving one-handed target acquisition on small touch-screens." In: *Proceedings of the working conference on Advanced visual interfaces*. 2008, pp. 146–153.

[94] Gabriel Ryan, Abigail Mosca, Remco Chang, and Eugene Wu. "At a glance: Pixel approximate entropy as a measure of line chart complexity." In: *IEEE transactions on visualization and computer graphics* 25.1 (2018), pp. 872–881.

[95] Takafumi Saito, Hiroko Nakamura Miyamura, Mitsuyoshi Yamamoto, Hiroki Saito, Yuka Hoshiya, and Takumi Kaseda. "Two-tone pseudo coloring: Compact visualization for one-dimensional data." In: *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.* IEEE. 2005, pp. 173–180.

[96] Léa Saviot, Frederik Brudy, and Steven Houben. "WRISTBAND. IO: expanding input and output spaces of a Smartwatch." In: *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 2017, pp. 2025–2033.

[97] Marcos Serrano, Khalad Hasan, Barrett Ens, Xing-Dong Yang, and Pourang Irani. "Smartwatches+ Head-Worn Displays: the" New" Smartphone." In: ACM SIGCHI. 2015.

[98] Marcos Serrano, Eric Lecolinet, and Yves Guiard. "Bezel-Tap gestures: quick activation of commands from sleep mode on tablets." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2013, pp. 3027–3036.

[99] Tomoki Shibata, Daniel Afergan, Danielle Kong, Beste F Yuksel, I Scott MacKenzie, and Robert JK Jacob. "DriftBoard: A panning-based text entry technique for ultra-small touchscreens." In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 2016, pp. 575–582.

[100] Ben Shneiderman. "Dynamic queries for visual information seeking." In: *IEEE software* 11.6 (1994), pp. 70–77.

[101] Katie A Siek, Yvonne Rogers, and Kay H Connelly. "Fat finger worries: how older and younger users physically interact with PDAs." In: *IFIP Conference on Human-Computer Interaction*. Springer. 2005, pp. 267–280.

[102] David Simkin and Reid Hastie. "An information-processing analysis of graph perception." In: *Journal of the American Statistical Association* 82.398 (1987), pp. 454–465.

[103] Gaganpreet Singh, William Delamare, and Pourang Irani. "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*. 2018, pp. 1–13.

[104] Srinath Sridhar, Anders Markussen, Antti Oulasvirta, Christian Theobalt, and Sebastian Boring. "WatchSense: On-and above-skin input sensing through a wearable depth sensor." In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2017, pp. 3891–3902.

[105] John Stasko, Richard Catrambone, Mark Guzdial, and Kevin McDonald. "An evaluation of space-filling information visualizations for depicting hierarchical structures." In: *International journal of human-computer studies* 53.5 (2000), pp. 663–694.

[106] Ashley Suh, Christopher Salgado, Mustafa Hajij, and Paul Rosen. "Topo-Lines: Topological Smoothing for Line Charts." In: *arXiv preprint arXiv:1906.09457* (2019).

[107] Juan Carlos Torrado Vidal, Germán Montoro, and Javier Gómez. "The potential of smartwatches for emotional self-regulation of people with autism spectrum disorder." In: (2016).

[108] Edward Tufte. *R.(2006) Beautiful evidence*. 2006.

[109] Jarke J Van Wijk and Huub Van de Wetering. "Cushion treemaps: Visualization of hierarchical information." In: *Proceedings 1999 IEEE Symposium on Information Visualization (InfoVis' 99)*. IEEE. 1999, pp. 73–78.

[110] Aku Visuri, Zhanna Sarsenbayeva, Niels van Berkel, Jorge Goncalves, Reza Rawassizadeh, Vassilis Kostakos, and Denzil Ferreira. "Quantifying sources and types of smartwatch usage sessions." In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2017, pp. 3569–3581.

[111] Daniel Vogel and Ravin Balakrishnan. "Occlusion-aware interfaces." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2010, pp. 263–272.

[112] Daniel Vogel and Patrick Baudisch. "Shift: a technique for operating pen-based interfaces using touch." In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 2007, pp. 657–666.

[113] Daniel Vogel and Géry Casiez. "Hand occlusion on a multi-touch tabletop." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2012, pp. 2307–2316.

[114] Daniel Vogel, Matthew Cudmore, Géry Casiez, Ravin Balakrishnan, and Liam Keliher. "Hand occlusion with tablet-sized direct pen input." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2009, pp. 557–566.

[115] Bryan Wang and Tovi Grossman. "BlyncSync: Enabling Multimodal Smartwatch Gestures with Synchronous Touch and Blink." In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI '20. Honolulu, HI, USA: Association for Computing Machinery, 2020, 1–14. ISBN:

9781450367080. DOI: 10.1145/3313831.3376132. URL: https://doi.org/10.1145/3313831.3376132.

[116] Taowei David Wang, Catherine Plaisant, Alexander J Quinn, Roman Stanchak, Shawn Murphy, and Ben Shneiderman. "Aligning temporal data by sentinel events: discovering patterns in electronic health records." In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 2008, pp. 457–466.

[117] Wei-Ching Wang, Chun-Ching Chen, and Tzu-Heng Chiu. "An Initial Exploration into the Design of Visualized Interfaces to Help Children Search for Books Using Smartwatches." In: (2017).

[118] Martin Wattenberg. "Visualizing the stock market." In: *CHI'99 extended abstracts on Human factors in computing systems*. 1999, pp. 188–189.

[119] Martin Wattenberg. "A note on space-filling visualizations and space-filling curves." In: *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.* IEEE. 2005, pp. 181–186.

[120] Marc Weber, Marc Alexa, and Wolfgang Müller. "Visualizing time-series on spirals." In: *Infovis*. Vol. 1. 2001, pp. 7–14.

[121] Dirk Wenig, Johannes Schöning, Alex Olwal, Mathias Oben, and Rainer Malaka. "Watchthru: Expanding smartwatch displays with mid-air visuals and wrist-worn augmented reality." In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2017, pp. 716–721.

[122] Vivian L West, David Borland, and W Ed Hammond. "Innovative information visualization of electronic health record data: a systematic review." In:

Journal of the American Medical Informatics Association 22.2 (2015), pp. 330–339.

[123] Pui Chung Wong, Kening Zhu, Xing-Dong Yang, and Hongbo Fu. "Exploring Eyes-free Bezel-initiated Swipe on Round Smartwatches." In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–11.

[124] Qinge Wu, Kelli Sum, and Dan Nathan-Roberts. "How fitness trackers facilitate health behavior change." In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 60. 1. SAGE Publications Sage CA: Los Angeles, CA. 2016, pp. 1068–1072.

[125] Haijun Xia, Tovi Grossman, and George Fitzmaurice. "NanoStylus: Enhancing input on ultra-small displays with a finger-mounted stylus." In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 2015, pp. 447–456.

[126] Robert Xiao, Teng Cao, Ning Guo, Jun Zhuo, Yang Zhang, and Chris Harrison. "LumiWatch: On-arm projected graphics and touch input." In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, pp. 1–11.

[127] Hui-Shyong Yeo, Juyoung Lee, Andrea Bianchi, and Aaron Quigley. "WatchMI: pressure touch, twist and pan gesture input on unmodified smartwatches." In: *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 2016, pp. 394–399.

[128] Xin Yi, Chun Yu, Weijie Xu, Xiaojun Bi, and Yuanchun Shi. "Compass: Rotational keyboard on non-touch smartwatches." In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2017, pp. 705–715.

[129] Yuk-ngang Zita Yip, Ze Zhu, and Yan-cheong Chan. "Reliability of wearable electronics—Case of water proof tests on smartwatch." In: *2017 IEEE 19th Electronics Packaging Technology Conference (EPTC)*. IEEE. 2017, pp. 1–5.

[130] Cheng Zhang, Junrui Yang, Caleb Southern, Thad E Starner, and Gregory D Abowd. "WatchOut: extending interactions on a smartwatch with inertial sensing." In: *Proceedings of the 2016 ACM International Symposium on Wearable Computers*. 2016, pp. 136–143.

[131] Junhan Zhou, Yang Zhang, Gierad Laput, and Chris Harrison. "AuraSense: enabling expressive around-smartwatch interactions with electric field sensing." In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. 2016, pp. 81–86.

Additional complimentary material has been included in the appendix section. Here are the questions used in the questionnaire of the second study of chapter five (5.3.6):

- **Q1:** What are the advantages of the Shift technique over two other techniques? (please explain)
- **Q2:** What are the advantages of Full-BezelGlide interaction technique over the Shift and Partial-BezelGlide techniques? (please explain)
- **Q3:** What are the advantages of Partial-BezelGlide interaction technique over the Shift and Full-BezelGlide techniques? (please explain)
- **Q4:** What didn't you like specifically about the Shift technique?
- **Q5:** What didn't you like specifically about Full-BezelGlide technique?
- **Q6:** What didn't you like specifically about Partial-BezelGlide technique?
- **Q7:** I prefer ..... technique over two other techniques.

The following questions were used in the second experiment of chapter six (6.3.5.4):

- **Q1:** Based on your experience from the experiment, which interaction area you prefer, large or small interaction area? (please explain)
- **Q2:** Which one of the layers (bands) on the interaction area, was the HARDEST layer to interact with, outer, middle, or the inner layer? (please explain)
- **Q3:** Which one of the layers (bands) on the interaction area, was the EASIEST layer to interact with, outer, middle, or the inner layer? (please explain)