CapCrown: Enhancing Smartwatch Interaction by Detecting Fingers when Rotating the Crown Using Capacitive Sensing

DAVID PETERSEN, Technische Hochschule Köln, Germany
MARVIN REUTER, Technische Hochschule Köln, Germany
MATTHIAS BÖHMER, Technische Hochschule Köln, Germany

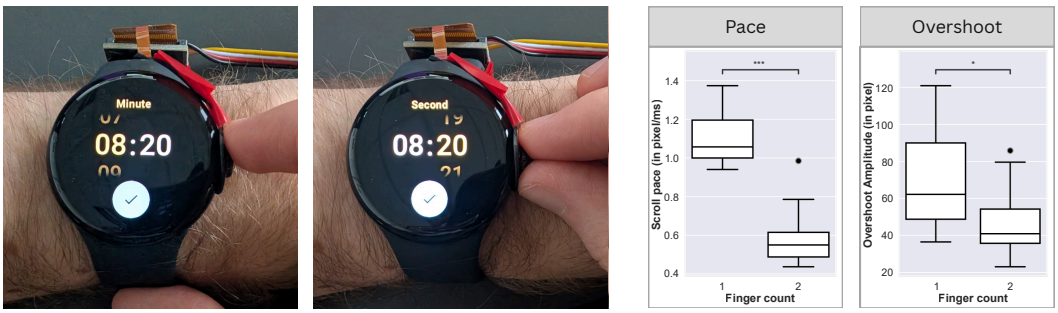


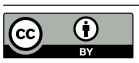
Fig. 1. Example of CapCrown (left pictures): a user can set the minutes of a timer with a one-finger rotation of the crown and directly alter the seconds with a two-finger rotation of the crown, without tapping on the display to change from minutes to seconds. Our study (right plots) reveals that scrolling with one finger is faster, while scrolling with two fingers is more accurate.

Smartwatch interaction is typically limited by the device's compact design and restricted input capabilities. Using the touch display occludes visual content, and for additional physical controls (e.g., buttons, bezel, crown) space is constrained. This paper enhances smartwatch interaction by expanding the functionality and thus increasing the input bandwidth of the crown. Initially, we investigate how users engage with a crown using two different numbers of fingers. Our informative study shows that crown rotation with one finger is faster compared to using two fingers. However, two-finger usage reduces the overshoot amplitude as well as number of fine-grained correction movements in a selection task. Subsequently, we develop a concept and a prototype that differentiates the number of fingers used in the crown interaction, which we call *CapCrown*. Finally, we discuss a set of applications that utilize this enhanced crown given the differences in interacting with one vs. two fingers. Our contributions include the study on crown interaction with different fingers, the sensor hardware, and the software using machine learning to detect fingers rotating a crown. Finally, we propose new smartwatch applications and system-wide functions that take advantage of our approach.

CCS Concepts: • **Human-centered computing** → **Interaction techniques**; **Empirical studies in ubiquitous and mobile computing**.

Additional Key Words and Phrases: Smartwatch, crown, input, multi-touch, study, capacitive sensing

Authors' Contact Information: David Petersen, Technische Hochschule Köln, Cologne, Germany, david.petersen@th-koeln.de; Marvin Reuter, Technische Hochschule Köln, Cologne, Germany, marvin.reuter@th-koeln.de; Matthias Böhmer, Technische Hochschule Köln, Cologne, Germany, matthias.boehmer@th-koeln.de.



ACM 2573-0142/2026/6-ARTEICS002
https://doi.org/10.1145/3812770

ACM Reference Format:
David Petersen, Marvin Reuter, and Matthias Böhmer. 2026. CapCrown: Enhancing Smartwatch Interaction by Detecting Fingers when Rotating the Crown Using Capacitive Sensing. *Proc. ACM Hum.-Comput. Interact.* 10, 4, Article EICS002 (June 2026), 22 pages. https://doi.org/10.1145/3812770

1 Introduction

Interaction with smartwatches is often constrained by their small form factor and limited input capabilities. On such small displays, traditional touch input suffers from the fat finger problem — users' fingers occlude the screen and make precise input difficult. While touch has been shown to be faster for scrolling tasks compared to using a bezel [11], it is slow for finding items on a map due to occluding the searched content [23].

Smartwatch crowns offer a promising and preferred [11] alternative, as they enable indirect, off-display input. Unlike bezel gestures or touch-based augmentation, crown interaction avoids screen occlusion entirely, since the crown is placed on the side of the watch from which the user touches it.

We set out to improve the interaction with smartwatches by enriching how people can use the crown. Inspired by Fitts' argument that users "can produce one bit of information per response per finger" [6] when interacting with a system, we set forth to increase the bandwidth when using the crown for interacting with smartwatch applications by taking into account how many fingers are used when rotating the crown.

The idea of leveraging finger count is already common in other interaction contexts. For example, multi-finger gestures are widely used on laptop trackpads and smartphone screens to distinguish between input types and control modalities. However, this principle has not yet been applied to smartwatch crowns, where space is limited but the fingers are already in contact with the input mechanism. Hence, the fingers could be sensed since they are touching the device or are very close to it.

Our approach is also motivated by the previous design of crowns for analogue watches: A mechanical crown can have multiple states through different extended positions. When pulled out to the first position, the crown has a different function. This enables the use of a single rotary crown to control different parameters per position (e.g. setting the hour, setting the minute, setting the date).

In this paper, we contribute a concept for incorporating finger count into crown interaction focusing on one-finger and two-finger crown rotation. We investigate how users can interact with the crown using one or two fingers by conducting a study. Our study shows differences in one-finger vs. two-finger crown interaction, with one finger being faster and two fingers being more accurate. For detecting finger usage we propose a capacitive sensing approach and develop appropriate hardware, hence the name of our approach: CapCrown. The data of our study was also used to train a machine learning model with an accuracy of 92 % for classifying finger count during crown interaction. Our findings further inform design recommendations for finger-aware crown interaction, as well as different application-specific and system-wide functionalities.

2 Related Work

Smartwatches usually have a limited input space because of the comparatively small touch screen and body. To address screen occlusion and the fat finger problem, researchers explored various approaches. These include different or novel input modalities like gestures and around-device interaction, as well as physical modifications to the device itself (e.g. custom cases or wristbands).

Gesture recognition allows the user to execute different types of functions based on different movements of the arm wearing the watch [9, 25] or the opposite hand [8, 27]. The gesture recognition

is often based on fine-grained sensor values that need to be processed in the cloud or on the connected smartphone due to computational demands or battery impact.

Around-device interaction [15] is useful, because the input can be extended beyond the small touch screen of a smartwatch. Han et al. [10] for example allow the user to begin an interaction on the touch screen and move beyond the screen to finish the interaction in-air. Additionally, the wearer's skin is used for around-device interaction [13, 22, 28]. However, around-device interaction setups are often not portable or require additional hardware like tracking systems, rings and styluses.

Utilizing the wristband of a smartwatch is another way of enabling input beyond the touch screen. WatchIt [18] is a prototype for a touch-enabled wristband. Funk et al. [7] developed a touch wristband for text input and BandSense [2] enables two point pressure touch on the wristband. Wristbands are essential for watches and thus a convenient way to improve the input space. However, because the band contacts various surfaces during regular, non-smartwatch use (e.g. wearer's body during sports), accidental input seems to be a significant limitation.

Xiao et al. [26] proposed a smartwatch prototype that allows panning, twisting, tilting and clicking the smartwatch case by mounting its display on top of two hall-effect joystick sensors. PressTact [4] utilizes four pressure sensors on the side of the watch that allow users to apply different levels of pressure to the watch used for scrolling, zooming and rotating. CaseTouch [24] is a prototype with capacitive stripes on the side of the case and on the upper part of the wristband that enables touch input without occluding the screen. Case-based input limits manufacturers in case design. For instance, actions like twisting require a square case. Others might require buttons or other additional hardware near or beneath the display, making the smartwatch overall bulkier.

Another approach to increase the input vocabulary on smartwatches is finger identification. MagTouch [17] uses a magnet ring and the built-in magnetometer of a smartwatch to distinguish between index, middle, and ring fingers based on the magnetic field and touch location. Similarly, SonarID [12] enables finger identification by analyzing sonar reflections during a touch using a deep learning model, achieving up to 93.7% accuracy. While both techniques enrich interaction by enabling finger-specific input on the same touchscreen area, they still involve direct contact with the screen and do not address the fat finger problem.

In addition to gestures, around-device interactions and modified hardware, physical rotation-based input like crown, bezel and wheel usage has been studied.

Neshati et al. [16] have shown that the use of optimal bezel regions (lower right) can result in 90+% screen visibility. In contrast, the crown is usually placed on the side of the watch body, resulting in no screen occlusion at all. In addition to that, Kerber et al. [11] have shown that crown interaction is preferred by participants for scrolling tasks. Also, crown usage was rated as having a lower physical demand compared to bezel usage. However, crown interaction is limited to one-dimensional tasks. Users can scroll upwards and downwards, but confirming a selection is always tied to additional input modalities, e.g. dwell, pressing a button or touching the screen. To increase the rotation-based input space, related work analyzed combining different rotation-based input [3], enhanced the bezel by increasing its degrees of freedom [14] or adding multi-touch elements [20], combined the crown with a trackball [23] or added joystick-like functionality to the crown [1]. However, modifying the crown itself by adding touch elements has not been studied yet.

In addition, the idea to multiplex rotation events with touch data has already been presented by Petersen et al. [19] on a rotary knob with a larger diameter. They demonstrated that varying numbers of fingers could be effectively combined with rotation to control different parameters using a single knob.

This is why we focus on enhancing the input space of a smartwatch crown by adding capacitive touch elements to differentiate between different numbers of fingers. Our contribution builds upon the work of Kerber et al. [11], showing that crowns are the preferred input method for scrolling tasks on smartwatches. We were inspired by the work of Brulé et al. [3] to extend the rotary input space, but without using multiple input devices. To achieve that, we incorporate the same state-of-the-art technology as Stanke et al. [24] and Reuter et al. [20] and adapt the concept of multi-touch on a rotary device of Petersen et al. [19].

3 Concept

The goal of our concept is to extend smartwatch interaction by rethinking how users physically interact with the crown. Instead of treating the crown as a one-dimensional input mechanism, we explore how the number of fingers used can serve as an additional input signal during interaction.

Traditional crown interaction is limited to linear control tasks such as scrolling, zooming or altering one specific value in one dimension. In contrast, other surfaces like touchscreen and trackpads support multi-finger gestures to differentiate between input types. Inspired by this, we propose to use finger count as an additional piece of information to enrich crown-based input.

The physical nature of the crown makes it well-suited for this kind of enhancement. Unlike touch interaction on the display, input via the crown avoids screen occlusion and thus mitigates the fat finger problem. At the same time, the fingers are already in contact with the crown during rotation, creating a natural opportunity to detect grasp-based variations without introducing new mechanical components, adding new interactive components or compromising the device's compact form factor by any other means.

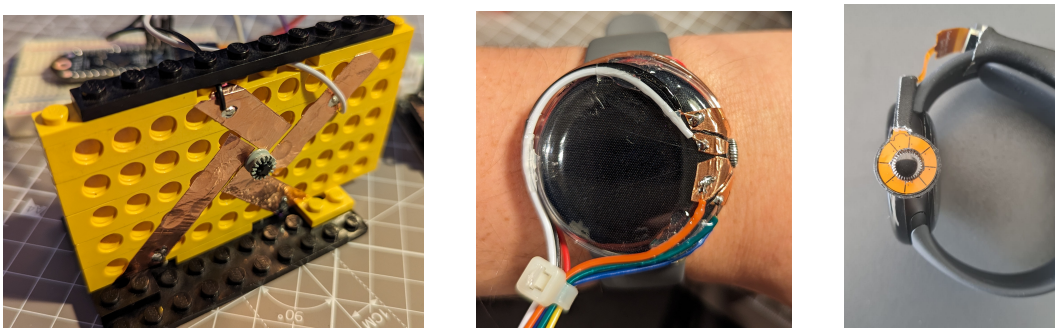
Kerber et al. suggest that digital crowns should be large enough to be easily graspable with multiple fingers [11]. Therefore, we did some informal pre-tests to learn how people touch the crowns of their watches (asking students and staff on our university campus). We found people rotating crowns with one finger as well as with two fingers. But we never found anybody who would naturally rotate their crown with three or even more fingers. This is likely due to the rather small diameter of crowns (about 6 mm on average for traditional watches [3]), and due to limited space between the smartwatch body and the wrist for easily placing a grasp with more than two fingers, such as a tripod or quadpod for precision grasps [5].

Our concept is based on the fact how the crown is grasped during rotation. By observing whether the crown is rotated with one or two fingers, the system can distinguish between two different interaction styles and map them to distinct functions. The concept is based on the principles of multi-touch input and the possibilities of analogue watches (e.g., mechanical crowns with multiple states), but reinterprets them in a digital, compact form.

To enable this, we propose a sensing approach based on capacitive input that detects the presence of fingers and the alignment of fingers around the crown. This approach does not interfere with the use of the crown and preserves its original functionality.

To realize this concept, we detect not only the rotation of the crown but also the presence and the spatial distribution of fingers around it. The finger position, whether along the side, rim, or spanning across the crown, serves as a cue for interaction style. For example, users may grip the crown with two fingers for more control, or rotate it with a single finger for speed.

Overall, we envision a grasp-aware crown that allows for richer interaction without modifying the physical interface. CapCrown does not alter the very nature of a smartwatch crown being a rotary encoder, but extends its functionality and increases the input bandwidth by adding one bit for one vs. two fingers.



(a) (b) (c)

Fig. 2. Evolution of CapCrown prototypes: a) first proof of concept using LEGO, b) copper tape attached to a watch housing, c) final prototype using custom flexible PCB.

4 CapCrown Prototype

To detect the number of fingers interacting with the smartwatch's crown, we employ capacitive sensing technology.

4.1 Evolution of the Capacitive Sensing

To investigate the feasibility of capacitive sensing for multi-finger crown interaction on smartwatches, we developed a series of hardware prototypes (see Figure 2). Each iteration progressively refined the sensing concept, exploring different form factors, material configurations, number of capacitive sensor pads, and integration strategies to balance signal quality, usability, and hardware constraints.

The initial prototype used copper tape segments around a replica consisting of a small rotating knob built with LEGO to evaluate the basic sensing principle (see Figure 2a). This setup allowed us to validate the concept and confirm that finger contact could be detected, since the array of four pads showed characteristic values for one-finger vs two-finger rotations of the tiny knob. Despite its rudimentary construction, this version served as a proof-of-concept.

To adapt this first concept to an actual application we adapted this construction to a smartwatch crown by using copper tape glued to a smartwatch cover (see Figure 2b). We doubled the number of pads to have eight capacitive measurements for more robust finger detection.

Compared to interaction concepts such as trackballs [23] or joysticks [1], our approach requires no additional mechanical components, since only passive electronic sensing is applied. As a result, the prototype benefits from a better durability and mechanical simplicity. Our design avoids moving parts, and is similar to CaseTouch [24], which relies on a touch-sensitive casing surface.

4.2 Final Capacitive Sensing Hardware

The final prototype utilizes eight capacitive pads arranged around the crown to detect capacitive changes (see Figure 2c). These pads are integrated into a custom-designed flexible printed circuit board (flex PCB)[1] that allows placement around the crown resulting in minimal influence on crown usage. A flexible (thermoplastic polyurethane) 3D print is placed between the smartwatch body and crown that holds the PCB in place. The flexibility of the PCB and the 3D printing allows the sensor

[1]Modified version of a bela Trill Flex sensor: https://learn.bela.io/using-trill/working-with-trill-flex/#designing-your-own-flex-sensor

to be pushed away without affecting the use of the crown. In addition, its dimensions are tailored to fit around the crown, ensuring consistent and reliable sensing performance. The capacitive sensing relies on self-capacitance, enabling individual detection per pad. Mutual-capacitance was not required, as the prototype only involves one-dimensional touch sensing around the crown, so ghosting is not an issue there.

The capacitive data is acquired using an ESP32 microcontroller, which reads the sensor values via the I2C protocol. The sensor values contain the capacitive values read from each pad. Communication between the smartwatch and the capacitive sensing is also handled by the microcontroller. Specifically, the ESP32 establishes a wireless access point (AP), facilitating data transmission via the User Datagram Protocol (UDP).

For our prototype we considered smartwatches with crowns, especially the Apple Watch and the Pixel Watch models. The crowns of the Apple Watch models are either very flat or even embedded in the watch's body, making two-finger interaction difficult. This is why we have chosen a Pixel Watch, which has a more protruding crown.

An off-the-shelf Pixel Watch 2 is used to communicate with the ESP32. The Pixel Watch is connected to the AP and an Android application running on the watch establishes a web-socket connection to the ESP to receive the raw capacitive values for all 8 touch pads. The application uses rotary scroll events from the Android API to be able to receive user induced crown rotation events. These rotation events and the capacitive values are multiplexed so that our prototype can distinguish different grasp pattern (e.g. one vs. two fingers).

Software and Hardware related specifications can be found on our GitHub repository[2].

5 Informative Study

To gain deeper insights into user behavior and assess the potential applications of CapCrown, we conducted an informative study. The study aimed to examine how people interact with the smartwatch crown when instructed to use one or two fingers. We chose the crown as the primary input device rather than the touchscreen to focus on extending an established input method rather than attempting to outperform the fastest known technique. This decision was informed by previous work showing that while touchscreens may offer higher speed for 1D scrolling, the crown remains the users' preferred device for such tasks [11].

Additionally, we aimed to determine how CapCrown could enhance interaction possibilities and whether it has any value for application design. These are our research questions:

- **RQ1**: How do timing and accuracy as well as task load differ when scrolling with one vs. two fingers?
- **RQ2**: How do transition times (from neutral position and from touching the crown to touching the touchscreen) differ when using one vs. two fingers?

How do participants touch the crown when operating with one vs. two fingers? The results of the study provide key insights into how multi-finger interactions can be leveraged for richer and more intuitive smartwatch functionalities.

5.1 Setup and Design

To investigate how users interact with a smartwatch crown using one or two fingers, we designed a within-subject study with two different types of tasks. Each subject was instructed to perform both task types using both one-finger and two-finger input, resulting in 4 different task sets per participant. The occurrence of task type and number of fingers were counter-balanced between participants.

[2]https://github.com/moxdlab/CapCrown

Before the actual study began, each subject completed a tutorial. In this phase, each type of task was explained and demonstrated, giving the participant time to practice and ask questions. No data was recorded during the tutorial and no time limits were imposed. Despite the number of fingers, the participants were not instructed on how to use the crown exactly. They were also not told to whether or how to touch the sensor, hence could ignore the sensor and naturally interact with the crown.

Each subject completed 30 tasks in the same order for each type of task and after finishing a block of 30 tasks for one task type, they filled out the NASA TLX questionnaire to assess the perceived task load.

The design of current smartwatch crowns can make two-finger access challenging, as they do not appear to be optimized for multi-finger use. While manufacturers can easily modify the crown's position, we cannot do this for our work using off-the-shelf smartwatches. Therefore, we used a 3D-printed elevation piece (10mm thickness) between the watch and the wrist to ensure easy one and two-finger use on the smartwatch crown. This also enables us to analyze the full crown usage around the cylindrical body, without limiting our analysis to the usually more accessible (e.g. upper) parts of the crown. Smartwatch manufacturers could place the crown closer to the top of the rim for better 2 finger reachability. We used a Google Pixel Watch 2 (display diameter: 41mm, crown diameter: 6.5mm).

In our study we excluded fling gestures. "Fling" describes the ability to quickly accelerate the crown for a short time, similar to snapping fingers with the crown in between. This gesture results in fast scrolling that continues over a small amount of time without the need to keep touching or rotating the crown. In our implementation the scrolling stops as soon as the physical rotation of the crown stops. This ensures that actual rotating events are captured without any fling-based scrolling physics.

Similar, we did not implement haptic feedback, as it would have introduced system-driven cues that could interfere with our goal of capturing raw interaction mechanics. Therefore, both fling gestures and haptic feedback were excluded to avoid masking differences between one- and two-finger input.

We included both a speed-focused and an accuracy-focused task to capture complementary aspects of crown interaction. The speed task allowed us to observe how quickly participants could scroll using each input technique without the influence of target-directed control. In contrast, the accuracy task introduced a clear goal, enabling us to examine how precisely participants could stop at a designated item.

Each task type was initiated using a different starting condition. The speed task began with a touch on the display, while the accuracy task started with a physical button press located next to the watch-wearing wrist. This distinction allowed us to measure different aspects of interaction timing—specifically, whether reaching out to the crown with one or two fingers results in different initiation times. By using both on-watch and off-watch starting conditions, we captured two types of reach: one involving interaction with the device itself, and one simulating activation from a neutral rest position.

5.1.1 Speed task. The goal of the speed task was to examine which interaction - using one or two fingers - enables faster interaction with the smartwatch crown. We abstracted real-world rotation tasks to gain insights into rotating performance for different numbers of fingers. In real-world scenarios, shortcuts or fling physics would be used (e.g., to reach the end of a list), but these do not provide insights into actual scrolling or rotation performance.

The interface for this task consisted of a countdown timer at the top and a scrollable list of numbers in the center of the display, see Figure 3b. Each trial began with a touch on the display,

(a) (b) (c)

Fig. 3. (a) Study setup consisting of a Google Pixel smartwatch, the CapCrown sensor attached to its crown, an ESP32 for collecting sensor data, and a small button for controlling the participant's hand in a resting position. (b) A user performing the speed task with two fingers on the crown. (c) A user performing the accuracy task with two fingers on the crown.

which triggered the start of a 10-second countdown. Within this 10-second time period, subjects were instructed to scroll through the list as fast as possible using either one or two fingers, depending on the current task type condition.

To account for potential directional bias, the scroll direction was alternated after each trial switching between upward and downward scrolling. This ensured that both directions were equally represented across input methods and subjects.

By this task design we aimed to get insights into how users perform when scrolling fast through a large list and which type of interaction (one vs. two fingers) may be more efficient when using the smartwatch crown.

5.1.2 Accuracy task. The accuracy task was designed to investigate how precisely subjects can control scrolling with the smartwatch crown using either one finger or two fingers. In this context, accuracy refers to how well subjects could scroll to a designated target item without overshooting or needing multiple scroll corrections during the interaction.

The interface displayed the target item at the top of the screen, a scrollable list of numbers in the center, and a confirmation button at the bottom (see Figure 3c). Each trial began with the subject pressing and holding a physical button for 10 seconds. This ensured that the subject's hand started in a relaxed neutral position before moving to interact with the smartwatch and also allowed us to measure the transition time from rest to interaction. The button was positioned directly adjacent to the wrist wearing the watch (see Figure 3a).

After releasing the button, subjects were instructed to scroll to the designated number and then confirm their selection with a touch on the display. Similar to the speed task, we alternated the scroll direction between trials (upward and downward scrolling) to eliminate directional bias across conditions. The scrolling distances were balanced across directions, ranging from 389.5 to 3874.5 pixels (mean: 2095.1 pixels). The target items were each 41 pixels in height.

Given the hardware specifications, 480 pixels could be scrolled per full rotation of the crown, corresponding to roughly 0.75° per pixel, or 1.33 pixels per degree of rotation.

5.1.3 Measurements. The within-subject study involved all participants to solve all tasks for all combinations of task type and number of finger. The number of finger (one or two) as well as the type of task (speed or accuracy) were the independent variables. While solving the tasks, the Pixel

Watch application persists timing, touch and sensor data: Each crown rotation results into a rotary input and gives the amount of pixels scrolled by these actions. Also, we keep track of each touch on the display. To identify finger interaction with the crown we used our sensor and persisted each incoming sensor data array.

This allows us to evaluate the dependent variables: scroll-pace (how fast did the participant scroll), rotation size and rotation count (how far did the crown rotate until the participant let go, and how often did the participant rotate, i.e. how often did the participant let go), time-on task (how long did it take to solve the accuracy task), overshoots (how far and how often has the goal been overshot in accuracy task).

Additionally, a camera setup was used that recorded the interaction with the prototype. This enabled a richer qualitative analysis, as interaction techniques could be identified. To assess the perceived workload, a NASA-TLX questionnaire was used.

5.2 Results

We conducted the study with 10 participants (6 male, 4 female), aged between 22 and 32 years (mean: 24.9, SD: 2.9). Nine participants were right-handed and two reported prior smartwatch experience. Eight participants wore the smartwatch on their left wrist and two on their right wrist, including the participant who reported having a dominant left hand.

In total, we collected data for 600 tasks per task type (30 tasks x 10 subjects x 2 input conditions), resulting in 1200 tasks across the speed and accuracy task types.

Each interaction was logged on the smartwatch, including rotary input, timing data, touch input and sensor data. To analyze the collected data we used a Jupyter Notebook environment. Time-based performance metrics were analyzed using the geometric mean to reduce the influence of skewed values [21] and the geometric standard deviation (GSD) is reported. For statistical comparisons, we applied paired t-tests, or Wilcoxon signed-rank test in cases where data were not normally distributed or contained extreme outliers.

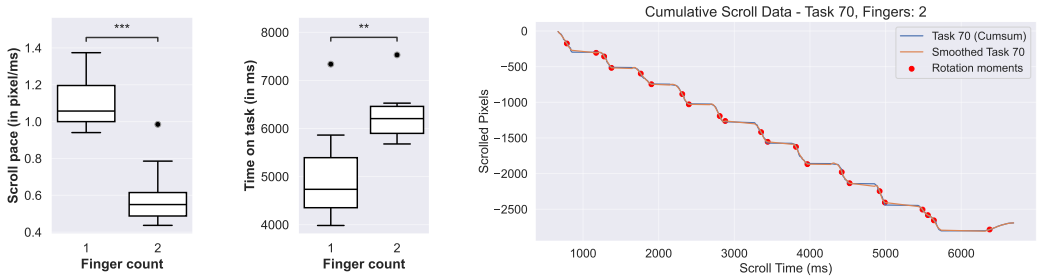
5.2.1 Speed Task. To evaluate the performance for the speed task type, we measured the average scroll pace in pixels per millisecond. Figure 4a shows the distribution of scroll speeds comparing one-finger and two-finger input conditions.

Overall, subjects scrolled significantly faster with one finger than with two fingers ($W = 0$, $n = 10$, $p = .002$). Participants on average scrolled 1.1 pixel/ms (GSD: 1.14) when using one finger. They reached 0.58 pixel/ms (GSD: 1.29) when using two fingers.

NASA TLX results are plotted in Figure 5. Our participants reported a significantly higher performance ($t(9) = -3.66$, $p = .005$) when using one finger while scrolling. Additionally, they experienced significantly higher frustration ($t(9) = -3.38$, $p = .008$) when using two fingers for fast scrolling.

5.2.2 Accuracy Task. To evaluate the accuracy task type, we will evaluate the time on task and different types of overshoots. When it comes to time on task, participants are significantly faster when using one finger ($W = 0$, $n = 10$, $p = .002$), see Figure 4b. Participants on average spent 4.95 seconds (GSD: 1.2 seconds) in the accuracy task when using one finger and 6.24 seconds (GSD: 1.09 seconds) when using two fingers.

To evaluate overshoots, we examined four different metrics: 1) overshoot accumulation, i.e., the total path lengths of all movements after the target is first reached; 2) overshoot amplitude, i.e., the maximum distance participants travel beyond the target; 3) number of fine-grained corrections, i.e. how often did participants passed the center of the target; 4) additional movement, i.e., the extra distance participants moved beyond the necessary amount. We need those four metrics, because the accumulated overshoot (1) alone does not imply assumptions about the maximum deviation from



(a) Scrolling pace. (b) Time on task. (c) Example of rotation moments evaluation.

Fig. 4.

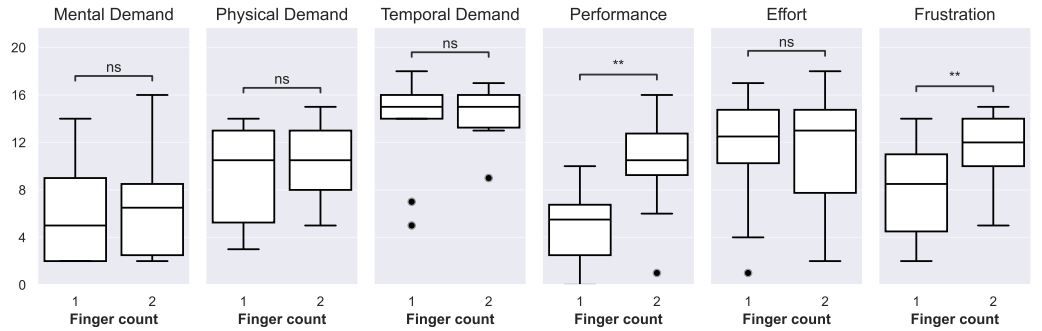


Fig. 5. NASA-TLX data for speed task.

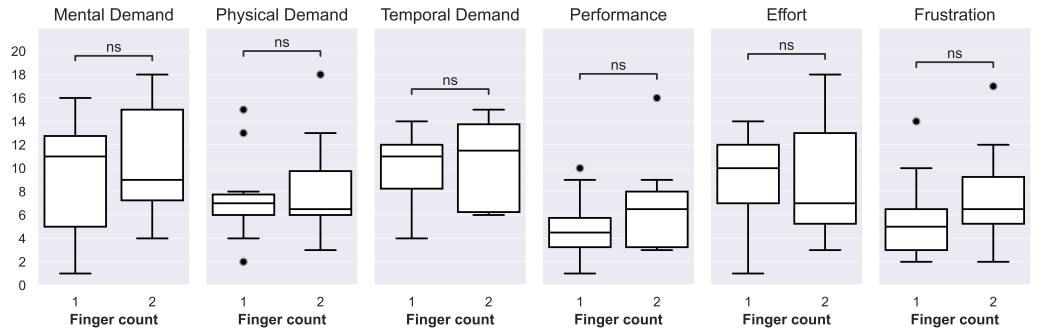


Fig. 6. NASA-TLX data for accuracy task.

the target (2) or number of fine-grained corrections within the target (3). Additional movement (4) is needed to evaluate unnecessary direction changes that are not necessarily caused by target overshooting, e.g. move in the wrong direction in the middle of the task.

In terms of accumulated overshoot (1), there are no significant differences between the number of fingers ($t(9) = 2.23$, $p = .052$, see Figure 7a). Participants using one finger overshot the target on

average by 141 (SD: 75) pixels and participants using two fingers overshot on average 85 (SD: 43) pixels.

However, when analyzing the overshoot amplitude (2), one finger has significantly higher values compared to two fingers ($t(9) = 2.91$, $p = .017$, see Figure 7b). Participants using one finger had an average overshoot amplitude of 69 (SD: 27) pixels and participants using two fingers 48 (SD: 19) pixels.

The analysis of number of fine-grained corrections (3) revealed that participants have significantly lower correction attempts when using two fingers, compared to using one finger ($W = 4$, $n = 10$, $p = .014$, see Figure 7c). On average, participants using one finger made 0.97 (SD: 0.19) corrections. Participants using two fingers made 0.8 (SD: 0.23) corrections.

In terms of additional movements (4), there are no significant differences between one and two finger usage ($W = 27$, $n = 10$, $p = 1.0$, see Figure 7d). Participants using one finger caused an average additional movement of 172 pixels (SD: 89 pixels), and those using two fingers caused an average additional movement of 217 pixels (SD: 209 pixels).

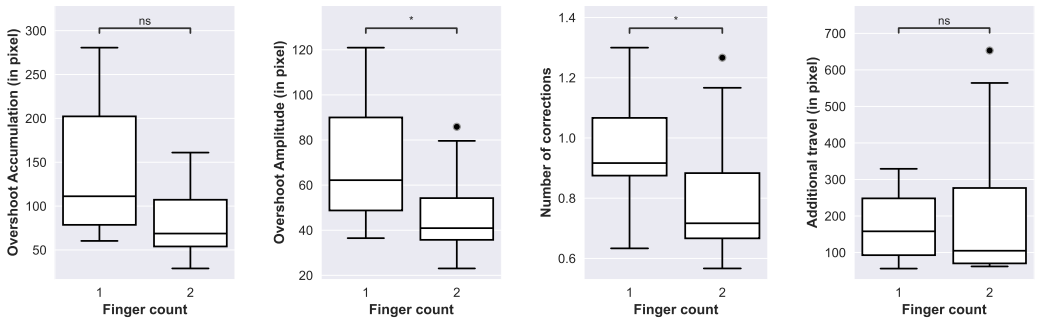
The NASA TLX results are plotted in Figure 6. There are no significant differences in task load metrics across both finger counts.

5.2.3 Moments of Rotation. To analyze how often participants paused during rotating the crown, we calculated turning points based on the scroll input (see Figure 4c). We define a moment of rotation (MoR) as a segment between two turning points i.e., a continuous scrolling movement without reversal or rest.

Across all tasks, the number of detected MoR events did not differ significantly between one finger and two finger input ($t(9) = 1.33$, $p = .217$, see Figure 8). Participants using one finger had an average of 23 MoR events (SD: 3.8) and participants using two fingers had an on average of 21 MoR events (SD: 2.86).

In the accuracy task, participants showed significantly more MoR events when using two fingers compared to one finger ($t(9) = -3.63$, $p = .0054$, see Figure 8). On average, participants had 9 MoR events (SD: 1.7) with one finger and 11 MoR events (SD: 1.81) with two fingers.

5.2.4 Switching between Input Modalities. The task design for the speed and accuracy task types involved different starting conditions. Participants had to press a physical button next to the watch before doing the accuracy tasks and use a digital button on the touch screen of the watch to start



(a) Overshoot accumulation. (b) Overshoot amplitude. (c) Number of fine-grained corrections. (d) Additional movement.

Fig. 7. Different metrics related to overshoots.

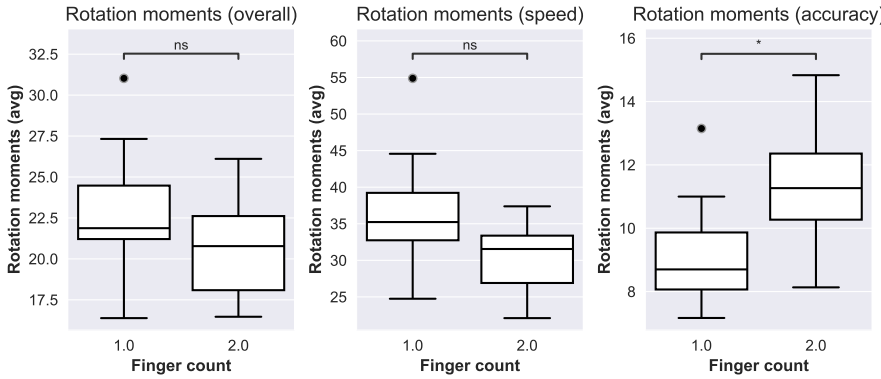


Fig. 8. Rotation moments divided in overall, speed and accuracy tasks.

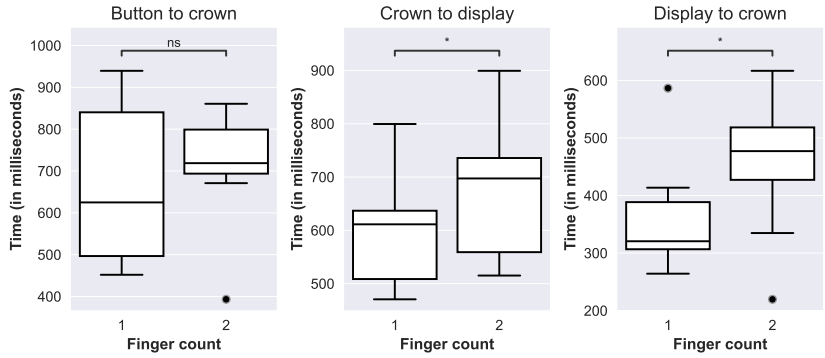


Fig. 9. Time for different actions while doing the tasks.

the speed tasks. This was done so that we are able to analyze time based data for different types of reaching the crown. In the following analysis, reaching out for the crown involves rotating the crown as well.

When reaching the crown from a neutral position that did not involve prior watch usage, there are no significant timing differences between using one or two fingers (W = 18, n = 10, p = .375). It took our participants on average 648 ms (GSD: 1.33) to reach out for the crown when using one finger and 702 ms (GSD: 1.25) when using two fingers.

When participants were previously interacting with the smartwatch, i.e. reaching out for the crown after touching the smartwatch screen, we found that participants are significantly faster (W = 4, n = 10, p = .0137) when using one instead of two fingers. It took our participants 349 ms (GSD: 1.26) to reach the crown using one finger and 444 ms (GSD: 1.35) when using two fingers. However, it must be noted that some participants used the thumb for the touch screen and the index finger for scrolling, resulting in a smaller transition time compared to having both fingers on the crown.

Finally, as the accuracy tasks needed to be submitted by using the touch screen, we were able to analyze differences between fingers when moving from crown to the touch display. We found that transitioning from crown to touchscreen is significantly faster (t(9) = -2.28, p = .0483) for one finger compared to having two fingers on the crown. It took our participants after the last scroll event

595 ms (GSD: 1.2) to touch the screen when having one finger and 662 ms (GSD: 1.2) when having two fingers on the crown.

5.2.5 Different Ways of Rotating the Crown. From analyzing the recorded video material, we were able to deduce seven different types of rotation techniques participants used while interacting with the crown. Additionally, the activation from each sensor is displayed in Figure 10. The plots are aligned such that pads facing north point towards the participant (similar to the display), and pads facing south point towards the participant's wrist.

Participants primarily used the index finger for one-finger interaction, with very rare thumb usage. Therefore, we focus on index finger interaction in the following analysis. Please note that one participant could use multiple different techniques during task solving.

top-to-bottom bursts: The index finger is placed parallel to the wristband. The fingertip is placed on pad 3 and then moves clockwise to pad 6 resulting in the fingertip touching the wrist. This small movement is repeated rapidly and looks like as if the participants are tapping on their wrist. This technique was used by 7 participants.

continuous rotation: The index finger is placed tangentially to the wristband. The fingertip touches any of the pads and is then moved in one long, continuous rotation around the crown. The fingertip keeps contact with the crown all the time. This technique works for clockwise and counter-clockwise rotations and looks like as if participants are using a miniature dialing disc. This technique was used by 2 participants.

full-finger sliding: The index finger is placed parallel to the wristband. The fingertip touches pad 1 to pad 4 and the stretched out finger is then slid over the crown resulting in a long clockwise rotation. This rotation involves all phalanges (phalanx distalis, phalanx media and phalanx proximalis) of the finger and looks like participants are pointing towards something. This technique was used by 3 participants.

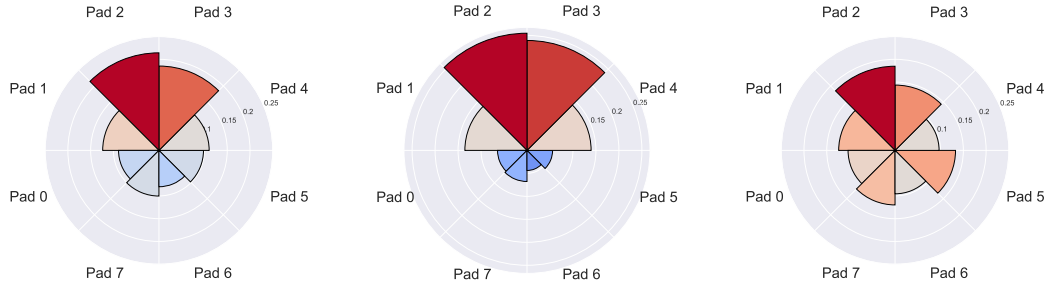
fingertip sliding: The index finger is placed like in *top-to-bottom bursts*, but only the third finger phalange (phalanx distalis) is used for the rotation. This results in smaller rotation values compared to *full-finger sliding*. This technique can be used for both clockwise and counter-clockwise rotations and looks like as if participants are swiping on a touch screen. This technique was used by 6 participants.

back-to-front bursts: The index finger is placed parallel to the wristband. The fingertip starts touching the upper part of the crown (pad 1 to pad 4) or the part that is facing away from the participant (pad 3 to 6). The finger is then rapidly pulled towards the participant or upwards, depending on the starting point of the finger. This interactions looks like as if participants were quickly scratching something. The interaction technique can be seen as the counterpart to *top-to-bottom bursts* for counter-clockwise rotations. This technique was used by 1 participant.

Figure 10b shows that most of the activation happened on pad 2 and pad 3 for one finger interaction. In total, the upper parts of the crown (pad 1, pad 2, pad 3 and pad 4) are activated most frequently, while the lower parts, which face the user's wrist, are used least. This indicates that techniques like full-finger sliding and fingertip sliding were used most and primarily at the top of the crown.

For two finger interaction, we identified two types of rotations. Both involve the use of the index finger and thumb:

tweezing: The tips of the index finger and thumb are very close and possibly touching each other while both fingernails touches the pads of the sensor or are very close. The crown is grasped in a tweezer-like manner. This technique was used by 3 participants.



(a) Crown usage over all tasks.

(b) Crown usage in accuracy task with 1 finger.

(c) Crown usage in accuracy task with 2 fingers.

Fig. 10. Activation of different pads during crown usage.

grasping: The index finger and thumb touch the crown with the finger pads, spreading the fingertips apart. The fingernails are not touching the crown or sensor. This technique was used by 9 participants.

Figure 10c shows that the pads are activated more distributed when using two fingers compared to one-finger use. This is expected, as both fingers need to be placed on the crown, and the thumb and index finger are probably often positioned opposite each other.

5.2.6 User Feedback. Nine out of ten participants provided open-ended feedback, offering additional insights into their experiences with one and two finger crown interaction across both task types.

Overall, seven participants expressed a clear preference for one finger input, frequently describing it as more natural, comfortable, and better suited for both fast and precise scrolling. In contrast, two finger input was often perceived as inconvenient due to the limited space around the crown and the physical presence of the surrounding sensor. Subject 10 noted that it was "a little hard to grab the crown with two fingers because of the sensor", while subject 5 stated: "The crown was too small for fast and effective two-finger use."

Participants commonly reported that one finger input felt more intuitive and allowed for smoother scrolling. Subject 3 noted: "Better than 2 fingers. More natural." Subject 1 shared: "It was easier to use with 1 instead of 2 fingers for me, because I could use the side of my finger instead of the tips."

In addition, two finger input was frequently associated with coordination issues and hand discomfort. Subject 7 described: "Hand cramps after a while because there's not enough space for the whole hand. After some time the crown feels harder to grip or harder to keep a consistent rhythm." Similarly, subject 8 noted that "fingers got in the way of each other."

Some participants described two finger input as allowing for finer adjustments. Subject 8 observed: "Usage was more accurate. It was easier to correct small mistakes (e.g. overshooting the target by 1 could easily be corrected)." Likewise, subject 5 mentioned: "It felt like you have better control over where to land."

6 Counting Fingers on the Crown using Machine Learning

The study involved using our prototype with one and two fingers for two types of tasks. While the sensor was not part of the study itself, it was already actively logging data during the task execution. This was done to gather knowledge about one and two finger use, without any prior assumptions about usage. Participants were not instructed how to use the crown, other than how many fingers should be used during rotations. This resulted in two classes of data, separated by one

and two fingers. Our participants were not explicitly instructed to touch the sensor while moving the crown, leading to data ambiguity. A simple algorithm like centroid detection can not reliably detect two fingers because participants might have had their fingers on the crown but only near the sensor. However, since our sensor reads raw capacitive values, we can detect finger presence even without direct sensor contact. We then utilized this raw data, combined with the two class labels, to train a neural network to infer whether users are touching the crown with one or two fingers.

6.1 Finger Count Classification

For processing the capacitive sensor data, we employ a neural network that interprets the read capacitive values and outputs the detected number of fingers. The model is implemented using TensorFlow Lite, a library optimized for microcontrollers, allowing efficient inference on the ESP32. Before being fed into the neural network, the capacitive values from the eight sensor pads are normalized to ensure a consistent input range.

Instructing our participants to use the crown with two fingers does not result in all data points having sensor values that exactly match two fingers. For example, when the participants rotate the crown with two fingers up to a point they cannot rotate further, they let go of the crown and grasp it again. In this process, one or zero fingers might have been near the sensor. The same is true for rotating with one finger. The re-grasping process is part of the data and was not removed. However, to make sure zero fingers are not overrepresented in the dataset, all data points containing no rotations were removed from training.

The neural network consists of a sequential model with seven fully connected layers. The input layer consists of eight neurons, corresponding to the eight capacitive pads. All hidden layers contain 1008 neurons in total that are activated using the ReLU function. This design expands the feature space, allowing the model to learn higher-level interactions between the sensor inputs. To prevent overfitting, three dropout layers were added after each of the first three hidden layers. The output layer consists of a single neuron with sigmoid activation function, providing a probability estimate for a binary classification task. This is particularly effective in determining whether a given capacitive signal corresponds to a single- or a dual-finger touch.

6.2 Training and Results

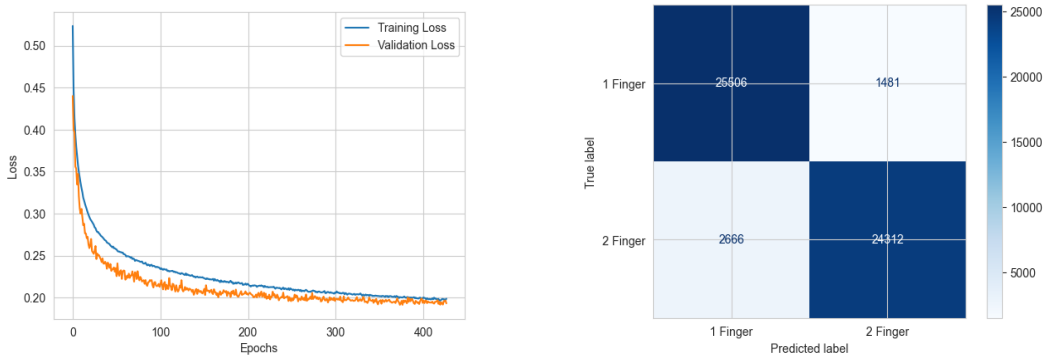
For the training, we separated the full classified dataset into two subsets randomly: 80% of the data is used for training and 20% is used for validating the model. We shuffled the dataset before training to eliminate order effects. In addition, we randomly shifted the sensor pad positions. This was done to make sure that the model learns the touch patterns independent of actual sensor positions. Without shifting, the model could falsely assume that upper crown regions indicated one-finger interaction and lower regions indicated two-finger interaction, disregarding actual sensor values.

The neural network model was trained on a MacBook Pro 2021 M1Pro chip on the CPU. With a learning rate of 0.001 and early stopping with a patience of 50 epochs and a batch size of 512, the model converged to a training loss of 0.1892 and a validation loss of 0.1934 after 422 epochs of training. The training and loss curve in Figure 11a show that the model is improving over time without overfitting on the training data, as the validation dataset accuracy improves as well.

The trained model achieves an accuracy of 92 % on the validation set, demonstrating decent performance in finger detection. The model reaches a precision of 92% and a recall of 92%, resulting in an f1-score of 92%. The confusion matrix is shown in Figure 11b.

6.3 Integrated Interactive Prototype

With this, we achieved the goal of combining the rotation events of the smartwatch crown with the touch events on the crown itself. With CapCrown, we can recognize how many fingers the user



(a) Training and validation loss curves.

(b) Confusion matrix for our trained model.

Fig. 11. Evaluation of our trained model.

places on the crown during interaction. The integrated final prototype can be used as an interactive system with live inference of the finger count during crown rotation.

We open-source the PCB around the crown, the ESP's firmware and the trained model. In addition, the raw dataset and the Jupyter Notebooks for training and analysis are accessible for other researchers to reproduce and use this sensor for future crown usage investigations.

7 Discussion

7.1 Lessons from Our Informative Study

Our study shows significant differences between one- and two-finger scrolling on a smartwatch crown. With one finger, participants rotated the crown faster as they reached higher scrolling speeds in the speed tasks. In addition, for the accuracy tasks, participants were overall completing the tasks faster using one finger to scroll. Furthermore, transitioning the focus of interaction between the crown and the touchscreen is faster when using one finger. When using two fingers, participants had less overshoot (in terms of amplitude) and fewer corrections for the accuracy tasks. For accuracy tasks, participants had significantly more moments of rotation using two fingers.

These findings suggest different characteristics of one-finger vs. two-finger crown interaction. These differences should be taken into account when enhancing smartwatch crown interaction by detecting fingers. These results are now being put into perspective.

When looking at speed-related measurements, techniques like full-finger slide could explain the differences we found. Using a stretched-out finger allows for longer scrolls and potentially faster scrolling speeds. This contrasts with the tweezing and grasping of two fingers, which requires constant re-grasping of the crown due to anatomical rotation limitations. This might explain the differences we found in both speed and moments of rotation and also why the participants rated the speed tasks more frustrating / performing worse when using two fingers.

The difference in transitioning between touchscreen and crown can be explained similarly. When using the index finger and the thumb for the crown rotation, at least one finger needs to move from the crown to the display. This is not the case for single finger interaction on the crown: When rotating the crown with only the index finger, the thumb can be used simultaneously on the touchscreen.

When it comes to accuracy, one finger rotations are overall faster in terms of time on task. This shows, that participants can reach higher speeds using one finger, while still maintaining a

certain accuracy, due to completing the task faster compared to using two fingers. However, the higher speed leads to a higher overshoot amplitude. In addition, participants also needed more fine-grained corrections around the target center when using one finger. We can conclude that two finger interaction on the crown is more accurate and can be used for more fine-grained control. This was also explicitly stated by two participants.

Based on these findings the following design recommendations for multi-finger crown interaction can be given. When using a technology like CapCrown and distinguish one-finger vs. two-finger to design different interactions, we recommend to

- use one-finger crown rotation for long scrolling tasks,
- use one-finger crown rotation for fast scrolling tasks,
- use one-finger crown rotation when fast switching between crown and display is required,
- use two-finger crown rotation for precise scrolling tasks,
- use two-finger crown rotation for error-critical tasks and high precision input.

7.2 Applications

As the state of the art, crowns of smartwatches have so far been restricted to one dimensional input and controlling exactly one specific parameter. Our work on CapCrown extends the interaction space of a crown by sensing the number of fingers being used during rotation. This gives rise to rethink and improve smartwatch interaction within specific applications, to introduce additional finger-specific gestures and to provide system-wide improvements. We made use of CapCrown and implemented sample applications for Android Wear OS.

7.2.1 Specific Applications. Rotary crown input is commonly used to navigate within lists or select items on menus. CapCrown gives rise to rethink such list navigation and menu selections. Navigating through long lists on a smartwatch can be tiresome. Scrolling acceleration methods like fling (as discussed earlier) can help to mitigate tedious scrolling through large lists. However, when using CapCrown, selection and navigation tasks can be improved beyond acceleration. Our *phone book application* distinguishes two different types of grasps while scrolling through a long list for an address book. When scrolling with two fingers, all contacts are scrolled in a linear manner, like in current smartwatches without finger recognition. In addition, when using one finger to scroll, the application quickly navigates the contact list by jumping between the contacts' initial letters. This way the friend John Doe can quickly be found by scrolling to letter J with one finger (long list), and then fine tuned to select the correct contact by using two fingers (precise scrolling). Such a strategy has evolved on larger rotary knobs [19].

Similarly, for a *map application* a two-dimensional navigation is wanted for panning and moving horizontally (East/West) and vertically (North/South). Using CapCrown, the movement in one direction could be done with one-finger rotations and in the other direction with two-finger rotation. This has similarly been addressed in CaseSense as an additional input control [24]. Further, navigation through large amounts of data or different visualizations, e.g., health applications, can be revised by using finger-aware crown interaction accordingly.

In other watch applications users need to switch between different parameters to set them, e.g., setting hours, minutes and seconds of an alarm clock. Setting a timer or alarm clock on many current smartwatches involves using the crown in combination with a touchscreen. Also, the crown itself can set parts of the timer, e.g. minutes, but cannot confirm or move between other parts, e.g. seconds.

As shown in Figure 1 with CapCrown one could directly control different digits with a different number of fingers (e.g., first digit: one finger; second digit: two fingers). Further, with CapCrown, we developed a *timer application*, that takes into account our design recommendations and allows

users to set hours, minutes, and seconds using only the crown, without needing to touch the screen: Rotating the crown with one finger results in setting one digit of the timer, e.g, hours. With two fingers the user will move the cursor to the next digit; i.e., move horizontally from hours to minutes to seconds and to the starting button. This enables the user to set multiple parts of the timer as well as starting it, without the need of touching the screen.

7.2.2 System-wide Crown Behavior. One challenge of having app-specific usage of CapCrown is that users will have to learn the two ways of interacting with the crown for each app individually. But the app themselves cannot easily communicate the one-finger vs. two-finger options and their corresponding functionality. Hence, the mapping of the finger-count is hard to learn. Another way is to provide more fundamental and system-wide functionalities, that behave similarly across different apps, but will be interpreted within the currently used app accordingly. For instance, the two-finger rotation could be used as a functionality for moving a cursor in a text, zooming in/out, pagination or controlling loudness. This would leave the one-finger interaction free for app specific functions like currently used. As such, one specific system-wide behavior can be used and different apps can interpret such gestures appropriately (like the pinching gesture on mobile apps is generally being used zooming gestures but will be implemented differently in apps like browsers, picture viewers, or maps).

7.2.3 Shortcuts within the System. Another option would be to keep one-finger based interaction as the common way of using all applications individually, but use the two-finger based rotation for specific system-wide shortcuts. For instance, *app switches* could be performed with two-finger rotations of the crown as a system-wide functionality. A two-finger rotation will open a menu with recent applications (like commonly used on desktops and smartphones). This mitigates the need to navigate to recent apps via a button and select the next app from the menu. Using the two-finger option appears to be reasonable for app switches, since users do not want to overshoot the target app, and the list usually will be rather short (no fast speed required). Other system-wide short cuts for two-finger rotations could be: controlling the current loudness, skipping to the next song on a music player, controlling display brightness, switching to the notification center and directly scroll through notifications, switching to phone book and directly scroll through contacts for a phone call.

CapCrown gives rise to different system-wide shortcuts. However, apparently only one shortcut can be active at a time. Users should be able to configure such behavior and adapt it to their own needs.

7.3 Limitations

One limitation of our trained ML-model is the sensitivity to noise and drift in the capacitive data. A slow build-up in sensor values - potentially caused by an electro-capacitive charging effect - reduced model performance. This phenomenon is caused by different capacitive values from person to person, requiring recalibration before and also during use of CapCrown. This did not have any major impact on our work since sensor was restarted and calibrated for every participant. A practical solution for future implementations would be to periodically recalculate a baseline and subtract low-frequency noise.

Furthermore, our findings are constrained by the exclusion of system features such as fling scrolling physics. These features can significantly alter the scrolling experience. Therefore, our recommendations apply specifically to scenarios where both one and two finger interaction are supported, and where raw mechanical input plays a central role rather than system-driven effects like fling.

Additionally, our study was conducted exclusively in a seated, controlled setting. We did not examine how different body postures or activity contexts (e.g., standing, walking, or running) may affect interaction performance or user behavior. As such, our findings may not fully reflect the range of conditions encountered in real-world, mobile scenarios.

One open question is on the transition time between one- and two-finger based rotations; i.e., how long does it take users to change from one grasp to the other? We could not measure this data in our informative study, since the technology we have built and trained based on the collected data is only now (i.e., after the study) available and able to tell apart one-finger and two-finger usage. That being said, however, our technical contribution now gives rise to answer this question and to conduct further studies of crown-based interaction designs. For instance, we could study micro gestures and take different touch points into account (e.g., rotation with one finger touching the crown at the top rim vs. the bottom rim). Also, mixed crown-display interactions become possible, with one finger touching the display while another one resting on the crown (which could not be sensed previously).

7.4 Threats to Validity

One item for discussion concerns generalization and soundness: Most current commercial smartwatches are considerably thinner than our prototype, due to the 10 mm elevation piece. However, as Brulé et al. [3] described, the positioning of crowns on watch bodies has already evolved over time. Hence, manufacturers of future smartwatches could also change the positioning of the crowns when integrating CapCrown. For instance, manufacturers could place the crown with enough distance to the skin or even tilt it on the bezel's rim (in a different angle) without necessarily affecting the overall smartwatch thickness. The space around the crown needs to be big enough for a two finger precision grip. All this needs to be evaluated in future development of smartwatch crowns to ensure our findings are generalizable for two-finger use on differently placed crowns and smartwatches with thinner bodies.

7.5 Future Work

One promising direction is the integration of sensing technology directly into the crown's mechanical structure, specifically into the pin of the crown itself. This would eliminate the need for the additional microcontroller and wiring, potentially reducing latency, noise and simplifying the overall hardware design. Another avenue involves more detailed interpretation of finger positioning during rotation. Future sensing approaches could capture these differences more accurately, e.g., by detecting rotation along the lower rim of the crown and use this data to enrich input classification.

Beyond distinguishing between one and two fingers, models could be trained to recognize different ways of rotating the crown (section 5.2.5). This would extend the interaction space of crowns even further by recognizing different grasp styles such as grasping or tweezing the crown to serve as distinct input modalities. Furthermore, the use of non-rotational gestures directly on the crown, such as tapping with multiple fingers, could further expand the expressive potential of crown-based input. For example, a three finger tap could be mapped to a shortcut, enabling richer interactions without requiring rotation.

To further improve the sensing performance, future iterations could increase the number of capacitive pads around the crown. A higher spatial resolution would allow for more precise detection of finger placement and motion patterns during rotation, thereby enhancing the model's ability to classify grasp styles. While the current implementation relies on self-capacitance, exploring mutual-capacitance sensing could offer benefits for more complex touch interactions with greater robustness. Additionally, employing time-dependent machine learning models may help capture dynamic aspects of interaction, such as grasp changes or finger repositioning over time.

The overall concept could also be extended beyond smartwatches to other small devices that feature rotatable knobs or crowns. With appropriate hardware specifications, it may even be possible to support more complex interactions such as three finger input, opening new design possibilities for miniature input surfaces. Similar work is already available for bezel-based smartwatch input [20]. A promising direction, alongside extending the context beyond smartwatches to other small devices, is to determine the minimal sufficient size of the crown (diameter, length). This investigation would establish the threshold for its effective usability.

8 Conclusion

In this paper we present CapCrown. CapCrown enhances crown-based smartwatch interaction by using capacitive sensing to detect fingers when a user is rotating a smartwatch crown. This piece of information can be used to double the interaction space and distinguish one-finger rotation vs. two-finger rotation.

To learn how people interact with the crown when using one vs. two fingers we first conducted an informative study. Our study shows that one finger interaction is faster for scrolling, and two finger interaction is more accurate in terms of the overshoot amplitude and the number of fine-grained corrections. This suggests that rotating the crown with one finger is better for tasks that need to cover a large distance quickly, and rotating with two fingers is better for tasks where higher accuracy is required. Based on the data we collected during the study, we trained a machine learning (ML) model to classify the number of fingers when somebody is interacting with the crown. We evaluated our ML-model to have an accuracy of 92 %. Based on the results of the study, we provided recommendations for the design of crown-based interactions. Further, we proposed how different applications can benefit from finger-aware interaction designs, especially when used as system-wide shortcuts.

All materials of our work are available as open source (layout for custom flexible PCB sensor, schematics, firmware for ESP, applications for smartwatch, study data, Jupyter notebook for informative study, Jupyter notebook for model training).

Acknowledgments

This work has been partly funded by the European Regional Development Fund under grant EFRE-20500002. We would like to thank Tjark Gaudich for his electrical engineering expertise and assistance with the PCB fabrication.

References

[1] David Ahlström, Khalad Hasan, Edward Lank, and Robert Liang. 2018. TiltCrown: Extending Input on a Smartwatch with a Tiltable Digital Crown. In *Proceedings of the 17th International Conference on Mobile and Ubiquitous Multimedia (MUM '18)*. Association for Computing Machinery, New York, NY, USA, 359–366. doi:10.1145/3282894.3289726

[2] Youngseok Ahn, Sungjae Hwang, HyunGook Yoon, Junghyeon Gim, and Jung-hee Ryu. 2015. BandSense: Pressure-sensitive Multi-touch Interaction on a Wristband. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI EA '15)*. Association for Computing Machinery, New York, NY, USA, 251–254. doi:10.1145/2702613.2725441

[3] Emeline Brulé, Gilles Bailly, Marcos Serrano, Marc Teyssier, and Samuel Huron. 2017. Investigating the Design Space of Smartwatches Combining Physical Rotary Inputs. In *Proceedings of the 29th Conference on l'Interaction Homme-Machine (IHM '17)*. Association for Computing Machinery, New York, NY, USA, 13–20. doi:10.1145/3132129.3132139

[4] Rajkumar Darbar, Prasanta Kr Sen, and Debasis Samanta. 2016. PressTact: Side Pressure-Based Input for Smartwatch Interaction. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI EA '16)*. Association for Computing Machinery, New York, NY, USA, 2431–2438. doi:10.1145/2851581.2892436

[5] Thomas Feix, Javier Romero, Heinz-Bodo Schmiedmayer, Aaron M. Dollar, and Danica Kragic. 2016. The GRASP Taxonomy of Human Grasp Types. *IEEE Transactions on Human-Machine Systems* 46, 1 (Feb. 2016), 66–77. doi:10.1109/

THMS.2015.2470657

[6] P. M. Fitts. 1954. The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Journal of Experimental Psychology* 47, 6 (1954), 381–391. doi:10.1037/h0055392

[7] Markus Funk, Alireza Sahami, Niels Henze, and Albrecht Schmidt. 2014. Using a touch-sensitive wristband for text entry on smart watches. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI EA '14)*. Association for Computing Machinery, New York, NY, USA, 2305–2310. doi:10.1145/2559206.2581143

[8] Hyunjae Gil, DoYoung Lee, Seunggyu Im, and Ian Oakley. 2017. TriTap: Identifying Finger Touches on Smartwatches. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. Association for Computing Machinery, New York, NY, USA, 3879–3890. doi:10.1145/3025453.3025561

[9] Jun Gong, Zheer Xu, Qifan Guo, Teddy Seyed, Xiang 'Anthony' Chen, Xiaojun Bi, and Xing-Dong Yang. 2018. WristText: One-handed Text Entry on Smartwatch Using Wrist Gestures. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–14. doi:10.1145/3173574.3173755

[10] Jaehyun Han, Sunggeun Ahn, Keunwoo Park, and Geehyuk Lee. 2017. Designing Touch Gestures Using the Space around the Smartwatch as Continuous Input Space. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces* (Brighton, United Kingdom) *(ISS '17)*. Association for Computing Machinery, New York, NY, USA, 210–219. doi:10.1145/3132272.3134134

[11] Frederic Kerber, Tobias Kiefer, Markus Löchtefeld, and Antonio Krüger. 2017. Investigating Current Techniques for Opposite-Hand Smartwatch Interaction. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '17)*. Association for Computing Machinery, New York, NY, USA, 1–12. doi:10.1145/3098279.3098542

[12] Jiwan Kim and Ian Oakley. 2022. SonarID: Using Sonar to Identify Fingers on a Smartwatch. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. Association for Computing Machinery, New York, NY, USA, 1–10. doi:10.1145/3491102.3501935

[13] Jiwan Kim, Jiwan Son, and Ian Oakley. 2025. Cross, Dwell, or Pinch: Designing and Evaluating Around-Device Selection Methods for Unmodified Smartwatches. doi:10.1145/3706598.3714308 arXiv:2503.02308 [cs].

[14] Kapil Kumar, Arindam Mondal, and Gaurav Gupta. 2019. Watch360: A Device to Enable and Detect Tilt, Translation and Rotation of a Watch Bezel. In *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE Press, Las Vegas, NV, USA, 1–6. doi:10.1109/CCNC.2019.8651875

[15] Marium-E-Jannat, Xing-Dong Yang, and Khalad Hasan. 2023. Around-device finger input on commodity smartwatches with learning guidance through discoverability. *International Journal of Human-Computer Studies* 179 (2023), 103105. doi:10.1016/j.ijhcs.2023.103105

[16] Ali Neshati, Bradley Rey, Ahmed Shariff Mohommed Faleel, Sandra Bardot, Celine Latulipe, and Pourang Irani. 2021. BezelGlide: Interacting with Graphs on Smartwatches with Minimal Screen Occlusion. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–13. doi:10.1145/3411764.3445201

[17] Keunwoo Park, Daehwa Kim, Seongkook Heo, and Geehyuk Lee. 2020. 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)*. Association for Computing Machinery, New York, NY, USA, 1–13. doi:10.1145/3313831.3376234

[18] Simon T. Perrault, Eric Lecolinet, James Eagan, and Yves Guiard. 2013. Watchit: simple gestures and eyes-free interaction for wristwatches and bracelets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 1451–1460. doi:10.1145/2470654.2466192

[19] David Petersen, Edgar Gellert, and Matthias Böhmer. 2023. Extending the Interaction Space of Rotary Knobs by Multi-touch-based Grasp Recognition. In *Proceedings of the 22nd International Conference on Mobile and Ubiquitous Multimedia (MUM '23)*. Association for Computing Machinery, New York, NY, USA, 198–209. doi:10.1145/3626705.3627797

[20] Marvin Reuter, Ali Ünal, Jan Felipe Kolodziejski Ribeiro, David Petersen, and Matthias Böhmer. 2025. MultiBezel: Adding Multi-Touch to a Smartwatch Bezel to Control Music. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA '25)*. Association for Computing Machinery, New York, NY, USA, Article 408, 6 pages. doi:10.1145/3706599.3720156

[21] Jeff Sauro and James R. Lewis. 2010. Average task times in usability tests: what to report?. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '10)*. Association for Computing Machinery, New York, NY, USA, 2347–2350. doi:10.1145/1753326.1753679

[22] Maximilian Schrapel, Florian Herzog, Steffen Ryll, and Michael Rohs. 2020. Watch my Painting: The Back of the Hand as a Drawing Space for Smartwatches. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '20)*. Association for Computing Machinery, New York, NY, USA, 1–10. doi:10.1145/3334480.3383040

[23] Dennis Stanke, Peer Schroth, and Michael Rohs. 2022. TrackballWatch: Trackball and Rotary Knob as a Non-Occluding Input Method for Smartwatches in Map Navigation Scenarios. *Proceedings of the ACM on Human-Computer Interaction* 6, MHCI (Sept. 2022), 199:1–199:14. doi:10.1145/3546734

[24] Dennis Stanke, Benjamin Simon, Sergej Löwen, and Michael Rohs. 2024. CaseTouch: Occlusion-Free Touch Input by adding a Thin Sensor Stripe to the Smartwatch Case. In *Proceedings of the 23rd International Conference on Mobile and Ubiquitous Multimedia (MUM '24)*. Association for Computing Machinery, New York, NY, USA, 172–183. doi:10.1145/3701571.3701583

[25] Hongyi Wen, Julian Ramos Rojas, and Anind K. Dey. 2016. Serendipity: Finger Gesture Recognition using an Off-the-Shelf Smartwatch. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 3847–3851. doi:10.1145/2858036.2858466

[26] Robert Xiao, Gierad Laput, and Chris Harrison. 2014. Expanding the input expressivity of smartwatches with mechanical pan, twist, tilt and click. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 193–196. doi:10.1145/2556288.2557017

[27] Cheng Zhang, Junrui Yang, Caleb Southern, Thad E. Starner, and Gregory D. Abowd. 2016. WatchOut: extending interactions on a smartwatch with inertial sensing. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers* (Heidelberg, Germany) *(ISWC '16)*. Association for Computing Machinery, New York, NY, USA, 136–143. doi:10.1145/2971763.2971775

[28] Yang Zhang, Junhan Zhou, Gierad Laput, and Chris Harrison. 2016. SkinTrack: Using the Body as an Electrical Waveguide for Continuous Finger Tracking on the Skin. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 1491–1503. doi:10.1145/2858036.2858082