sensing, interaction & perception lab

Department of Computer Science, ETH Zürich

Bachelor's Thesis

On-body touch input in VR using wrist-watch touch detection

Markus Pobitzer

1st supervisor

Andreas Fender
Sensing, Interaction & Perception Lab
ETH Zürich

2nd supervisor

Prof. Dr. Christian Holz
Sensing, Interaction & Perception Lab
ETH Zürich

August 18th, 2021

Markus Pobitzer
On-body touch input in VR using wrist-watch touch detection
Bachelor's Thesis, August 18th, 2021
Supervisors: Andreas Fender and Prof. Dr. Christian Holz

ETH Zürich
Department of Computer Science, ETH Zürich
Universitätstrasse 6
8092 Zürich

On-body touch input in VR using wrist-watch touch detection

Markus Pobitzer
Sensing, Interaction & Perception Lab
Department of Computer Science, ETH Zürich
pobmarku@student.ethz.ch

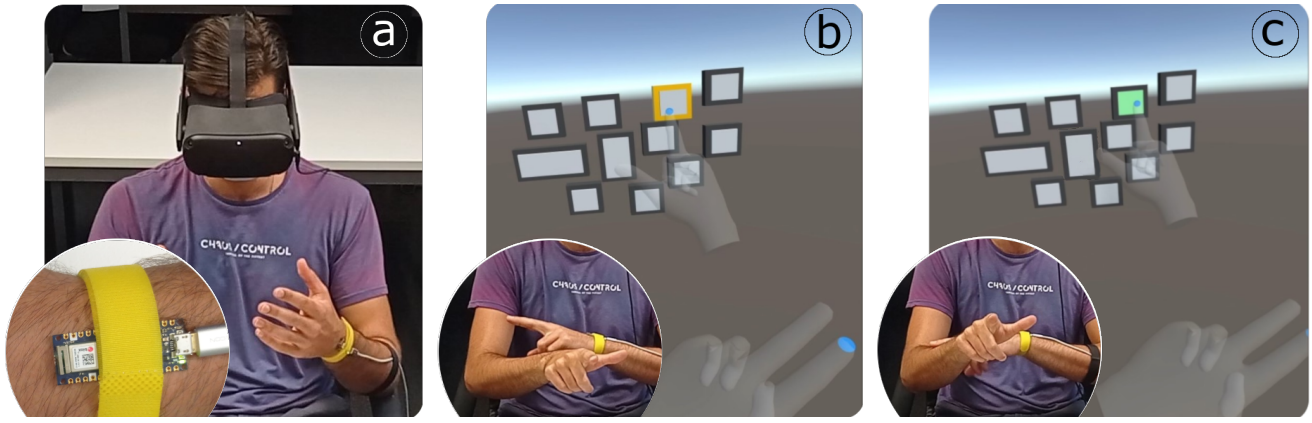

Figure 1. We present a method that detects on body touches through a wrist worn sensor. We combine these touch events with finger tracking to activate menus in a virtual environment. (a) The user is wearing an Arduino and a head mounted display. Through the display, virtual buttons can be seen (b) and the highlighted button gets activated (c) and turns green. The activation of the button is a combination between finger tracking and on body touch.

ABSTRACT

Modern interaction methods in virtual reality (VR) are based on controllers or in-air gestures. This overlooks smartwatches and wearables strapped on the wrist as a complement to in-air gestures. In this thesis, we quantify the effect of haptic on-body input in VR on accuracy and user experience for menu item selection. We combine the finger tracking of a head-mounted display (HMD) with a wrist-worn inertial measurement unit (IMU). A touch interaction corresponds to a peak in the acceleration data of the IMU. For the evaluation, we conducted a user study consisting of two parts and four participants. We provide an interaction method in VR using a low-cost IMU worn on the wrist. Our method is comparable to state-of-the-art interaction methods regarding completion time.

Keywords

Human-centered computing; Human computer interaction (HCI); Interaction techniques

INTRODUCTION

Contemporary mixed reality systems use controllers (e.g., Oculus Quest 2 [6] and HTC Vive [7]) or in air gesture recognition (e.g., Microsof HoloLens 2 [8]) for interactions in mixed reality. Often, they support both as seen with the Oculus Quest series [5].

Positional tracking is a fundamental part of VR. Mainly controllers were used to track the position of the hand in space. Nowadays, finger tracking is a viable option too to track hand and even fingers accurately [27]. One of the big advantages of the controller is the variety in interaction possibilities through the buttons. But the assignments of all the buttons need to be learned by the user and changes from application to application.

The use of a controller can cause a break in the experience of virtual reality. Representing the position of hands in VR comes down to using a model of a controller or stiff hand models. It is not possible to represent the intricate finger gestures and positions of the real world or to grab things with the fingers since the controller must be held. But these fine interactions are often necessary for enterprise applications e.g., manufacturing processes and augmented reality systems where a controller is a hindrance to real-world tasks.

Interacting in VR with the own fingers can provide a more natural experience for the user. Finger tracking faces still some challenges such as occlusion and inaccuracies when it comes to the exact differentiation of a touch event. This can lead to problems when interacting with the own body. These challenges mainly arise from the positioning of the HMD and the sensors used for finger tracking, leading to inaccuracies in depth sensing.

To make the finger tracking more versatile, we combine it with a wrist-worn IMU to detect peaks in the acceleration of the hand. The peaks are caused by on-body touches and registered as a touch event like a button press. Peaks get detected by a rate-of-change score (RCS) based on previous acceleration values and the absolute difference between consecutive values of each axis. Touching the own body provides additional haptic feedback. In combination with finger tracking, the touch event can be used to activate different user interface (UI) elements. Similar IMUs as we use can be found in smartwatches, which are becoming increasingly commonplace. No additional controller is needed, and our method does not suffer from occlusion.

To see if our method is comparable to state-of-the-art interaction methods, we conducted a study comparing different UI item selection methods with our method. In detail, we looked at the completion time of different tasks, error rate, and user feedback. We collected the user feedback through a questionnaire.

The study shows that low-cost IMUs can be used for on-body touch interactions. The completion time regarding a task with our method is comparable to currently used interaction methods. The questionnaire showed a high acceptance of our method.

We conclude that detecting touch interactions is possible with a wrist-worn IMU and a relatively simple peak detection method such as the used RCS method. Through the combination of touch detection and finger tracking our method avoids problems with occlusion when detecting on body touches. Our method naturally provides haptic feedback. No additional handheld controller is needed.

The work in this thesis makes the following contributions.

- Introduce a method for sensing on body touches with a simple wrist-worn accelerometer and show how it can be used to select menu items in VR.
- Show accuracy, error rate, and liking of our method compared to other methods through a subjective rating and a questionnaire with 4 participants.

We compare our method to different state-of-the-art interaction methods such as button activation by in-air poking that relies only on finger tracking, UI element activations by ray cast with a controller, and pinching a combination of ray cast with in-air gestures.

RELATED WORK

Our work is related to surface interaction with wearable sensors in combination with VR. Our work is closely related to methods that use the body as an input surface or an accelerometer as the wearable sensor.

The most relevant work is TapId [24] that uses wrist-worn IMUs to detect which finger interacts with a surface. Similarly, there are approaches to wear the IMU as a ring on the finger [13]. Wrist-worn devices are found in ViBand [21] to detect interactions with objects via an accelerometer, ActiTouch [30] that uses the body as a radio frequency (RF) waveguide to detect touches and Skinput [16] that uses mechanical vibrations to detect touches on different locations on the arm. OmniTouch [15] showed that depth sensors can be used for interactions with various surfaces.

Touch interaction with Accelerometers

IMUs are built in a lot of different devices such as mobile phones and smartwatches. Using the accelerometer data from the IMU as a form of input is not new. There is work about gesture recognition through accelerometer [22, 10]. Gesture recognition is not our focus here and can also be compensated well with finger tracking since the tracking methods give us detailed information of the finger positions. Oculus Quest already recognizes some basic gestures with its hand-tracking [2].

Using an accelerometer with a high sampling frequency opens a lot of possibilities to interact with surfaces and objects as shown with ViBand. The used frequency of the accelerometer signal was 4000Hz compared to 100Hz of common accelerometer signals. This allowed to better distinguish different vibrations even leading to hand gestures and on-body interactions to the point where the position of the interaction was classifiable from the signal alone. The focus of our work is to use a smartwatch in combination with an HMD and VR.

TapId combines VR and an accelerometer to enable rapid touch interactions in VR. Not only were the authors able to detect tap interactions with surfaces from the accelerometer signal but also the specific finger that interacted. The focus was to interact with a surface such as a table to reduce fatigue. Interactions with the own body were also mentioned but not carried out in an example. We decided to use the described tap event detection in TapId to explore on-body touch interactions.

Another way to detect which finger is currently interacting with a surface is to strap the accelerometer to the corresponding finger as shown by Gu et al. [13].

The mentioned works have shown that it is possible to detect touches on surfaces and the own body with a worn IMU. The applications in the mentioned works ranged from detecting touch positions on the arm, distinguishing which finger is involved in the touch event up to complex input methods to work in VR without a keyboard and mouse. This shows that touch interaction can be used in different scenarios and the use of other methods such as a classifier or hand tracking goes hand in hand.

On-body touch input

There are various other ways to detect touch inputs on the body and use it as an input medium. Noteworthy options include depth cameras [14, 15, 12], Skinput [16], ActiTouch [30], skin-worn sensors [29, 23, 18, 25], implanted user interfaces [17], photoelectric sensors [20, 26] and piezoelectric sensors [11, 19].

PalmRc [12] shows that with depth cameras it is possible to detect the non-dominant hand and recognize touch events with the dominant hand. The depth camera in PalmRc is stationary whereas OmniTouch [15] shows that interactions on surfaces, especially the arm are possible with a wearable

depth camera. Imaginary phone [14] is another example where a depth camera is used to create on-body touch interactions. In the mentioned works here the UI elements were all on the body whereas we have our focus more on the activation of menu items that are in midair, and we use the on-body touch interaction as a trigger to activate selected elements.

Skinput presents a way to use the human body as an acoustic transmitter. Their custom-build armband collects bio-acoustic signals where they can detect taps on different positions on the arm. A custom-made armband is also used with ActiTouch. It uses the body as an RF waveguide and detects touches by varying signals through electrodes. The authors combine the armband with a VR system to incorporate touch events with spatial tracking of the fingers. Not only taps get detected, but it is also possible to detect continual contact from the RF signal alone seen in the drawing application they present.

A completely different approach to body touch input is to use skin-worn sensors. Such sensor overlays are thin and flexible and can detect touch inputs [29]. Similar input forms come also in form of visually appealing patterns such as tattoos [23] and other appealing forms with a lot of coupled potentials such as NFC tags and visual feedback in the form of thermochromic displays [18]. Further research shows that we can print the input sensors that provide multi-touch inputs [25]. All mentioned overlays can bend and stretch to a certain degree to fully accommodate the change in the skin.

Previous work has highlighted different ways of on-body interaction often using specific hardware. The method in this thesis uses a common IMU, nowadays found in a smartwatch in combination with finger tracking. Specifically looking at on-body touch interactions that have been overlooked or often only mentioned in similar work.

IMPLEMENTATION

Our method relies on a device that contains an Inertial Measurement Unit (IMU) worn on the non-dominant hand. The built-in accelerometer provides us with acceleration data when the hand moves. In our work we use an Arduino Nano 33 IoT [1] connected over a serial port. We use a Dell XPS 15 9575 to evaluate the accelerometer data and detect a tap event. We combine this with a head-mounted display (HMD) and finger tracking. First, we will discuss how to detect a tap event. Then, we discuss the incorporation of the tap event into VR.

Detecting Tap events

The accelerometer that is built into the Arduino is an LSM6DS3 and has a sampling frequency of 104Hz [9]. It sends a value for the acceleration of the three-axis x, y, z. based on this data we detect if the hand with the sensor on taps, or gets tapped by the other hand. We do not distinguish the form of interaction, only if a tap event occurs or not. If we look at Figure 2 we see how sensor data looks like for such tap events. From this, on we use a method called rate-of-change score (RCS) [24] to distinguish between normal hand movements and tapping events. The main difference is that hand movements have not as an abrupt change in the acceleration as the taps. To calculate the new RCS value we use the last

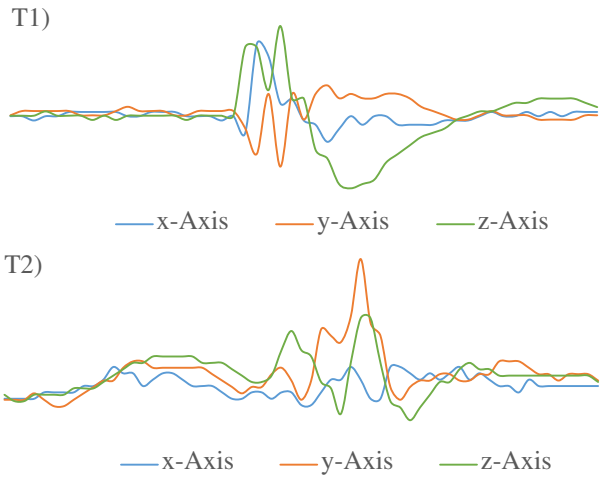

Figure 2. Two different acceleration data visualized at a tap event. The values for each axis are centered. In T1) we see the values of the three axis when the main hand taps the palm of the hand where the IMU is on. In T2) we see the values when the hand with the IMU on taps the main arm.

RCS value and the absolute differences in the values of each axis (VA).

$$RCS_t = \frac{RCS_{t-1}}{1.6} + \sum_{Axis} |VA_t - VA_{t-1}| \quad (1)$$

A tap event gets noticed after the current RCS value exceeds a threshold of 0.35. This threshold has been chosen through observations of tap events and tweaking. After each successful detection of a tap, we reset the current RCS value to 0 and wait for a backoff time of 300ms. Therefore, it is possible to detect up to 200 tap events per minute. We take these actions to not detect the same tap event more than once. The tap gets propagated to the VR application via an event.

Incorporating in VR

To use the tap event in a virtual reality environment we incorporate an HMD with finger tracking. The combination of the tap event and the virtual environment opens new ways of interacting with UI elements such as buttons. Especially useful are the positions of the finger and hand provided by the tracking system. We use the tip of the index fingers as the main point of contact for interactions with UI elements. If an interaction with elements far away is necessary, we use a ray starting from the hand. Such contact points are necessary to distinguish if a trigger of a tap event activates a currently selected UI element or if it gets ignored. Buttons have therefore colliders that the tip of the index finger can collide with and set the attached button as the currently selected. When a tap event arrives, the application knows exactly which button to activate.
A tap event can be generated when the hand with the IMU sensor touches an object or gets touched by the other hand. We specifically looked at taps that are created by touching the own body. This also means that the place where touch interaction happens does not need to be in the tracking field of the HMD nor is there a problem if it is occluded as long as we use the other hand to select the UI elements. This enables us to get a

more precise tap event if the UI elements are on the own hand compared to finger tracking alone that suffers from occlusion if the two hands interact with each other.

In our work we use an Oculus Quest that comes with a 6DOF inside-out tracking through 4 built-in cameras [3]. The inside-out tracking is used to get the current position of the user's fingers and hands. We use Unity [28] and C# to incorporate the different aspects of our program.

UI elements

We created two different UI elements. A button in the reach of the hand and a panel that is outside of the hands reach and needs a ray for interaction. Both UI elements have three states. Starting with the default state the element is not highlighted and gets ignored if a tap event arrives. The element changes color to blue in the selected state, to give the user visual feedback, which element is currently selected. If a tap event arrives and the element is selected then it transitions to the activated state. The activated state changes the color to green to better show the activation. From the activated state we can transition to the selected and default state. For our study, we gave each element a border that can be highlighted such that the user knows which element to activate.
In the study each UI element can get activated through different means such as poking a button in midair, using a controller, or the hand combined with a pinching gesture. Here we only describe the activation of the elements through our method because we used common practice techniques for the other methods.

To make the button selection possible we gave it a collider that sticks out of the button. We use the tip of each index finger as the point of collision. A sphere scaled to the dimensions of the finger and able to collide with objects follows the exact position of the fingertip. The button transitions from the default state to the selected state whenever such a sphere collides with it. It leaves the selected state only when a tap event arrives, or the sphere leaves the collider of the button.

The panels are positioned outside of the reach for the users. Instead of a sphere to collide with we use a ray that starts from the user's hand. To derive the pointing direction of the hand we use Oculus PointerPose [4]. It gives us the starting point and direction of the ray we want to use. In addition, it is compliant with other apps provided by Oculus. The pointing direction is calculated by the Oculus software. Once we got the direction, we can cast a ray from the starting point and see what objects the ray hit. If it is a panel, we set it to the selected state. The selection process follows the same principles as the button. As long as the ray hits the panel we stay in the selected state and leave only if a tap event arrives or the ray misses the panel.

STUDY

To better understand if the on-body touch input is a viable option for VR input methods we conducted a within-subject study. In the study, we compare different state-of-the-art technologies with our method. The study is split up into two blocks and each block has different UI elements that can be activated based on a certain condition. The UI elements in the first Block are buttons and the UI elements in the second block are panels. In each block, we have four different UI groups, whereas we use the first UI group as training. Per condition, there are 130 activations of UI elements (including the Training group), whereas we take 90 activations of elements (the Training group excluded) into consideration for the evaluation of the study. Over the two blocks we evaluate five conditions with 450 activations per participant, Training group excluded.

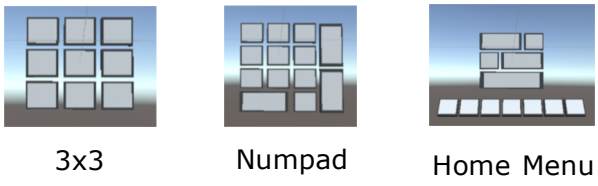

Figure 3. Block 1 with the three relevant UI element groups 3x3, Numpad and Home Menu

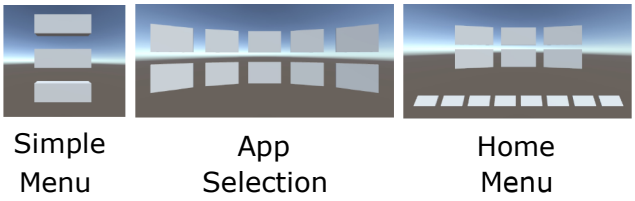

Figure 4. Block 2 with the three relevant UI element groups Simple Menu, App Selection and Home Menu

Participants

We invited four participants (2 female, 2 male, mean age 24.75, all right-handed) to go through the study. Due to technical issues, we excluded one participant from block 2.

Apparatus

The study was conducted on a Dell XPS 15 9575, with an Nvidia GTX 1050 ti as an external graphics card. The used HMD is an Oculus Quest with the corresponding Oculus touch controllers. For the IMU we used an Arduino Nano 33 IoT connected via USB over a serial connection to the laptop. We used a tablet to fill out a questionnaire.

Task

Each participant must activate highlighted UI elements from a certain element group under a condition. Activating the highlighted element gives positive visual and audio feedback. Activating another element gives negative visual and audio feedback. In the first block, the elements are buttons to press and in the second block, the elements are panels to activate. The experimenter instructed to activate only the highlighted elements as fast as possible.

In the study, we define the performance as the completion time of a UI element group under a certain condition. We keep track of how often the participants activated a wrong UI element i.e., not highlighted, and count it as an error. With questionnaires after each finished condition, we want to find out how the participants responded in correspondence to different UX aspects in the run.

Figure 5. Representation of the Task and visual feedback from Block 1, 3x3. In A we see the highlighted button, in B when the index finger hovers over a button, in C the right button is pressed and the next button gets highlighted and in D a wrong button is pressed.

Procedure

A participant is sitting on a chair in a comfortable position. On the wrist of the non-dominant hand, the participant wears the Arduino with the IMU on it in a tight but comfortable way. During the completion of a task, the participant wears the HMD and takes it off to fill out the evaluation after each condition.
Starting with the first block and then going on to the second block. For counterbalance, we randomized the order of the conditions in which the participants had to complete the task. It took a participant around 40 minutes to complete the study.

Block 1

In the first block, we named the UI groups Training, 3x3, Numpad, and Home Menu as shown in Figure 3. The UI group consists of buttons that can be activated based on the current condition. The block has two conditions named In Air Poking and On-Body Touch.

In the first condition (In Air poking) a button gets selected if the index finger of the main hand hovers above it and it gets activated when pressed down. In the second condition (On-Body touch) a button gets selected if the index finger of the main hand hovers above it and the selected button gets activated when a tap event arrives.

Block 2

In the second block, we named the UI groups Training, Simple Menu, App Selection, and Home Menu as shown in Figure 4. The UI group consists of panels that can be selected when hit by a ray cast and activated based on the current condition. The block has three conditions named Controller, On-Body Touch, and Pinching.

In the first condition (Controller) we use the Oculus touch controller to create the ray to select the panels and activate a selected panel through the "PrimaryIndexTrigger" on the controller. With the second condition (On-Body Touch) we create a ray from the tracked hand to select the panels. A selected panel gets activated when a tap event arrives. In the third condition (Pinching) we select the panels in the same way as with the second condition and activate a selected panel by making a pinching gesture (closing the thumb and index finger).

Hypotheses

We hypothesize that our method is comparable regarding the completion time of the tasks except for the condition Controller. We think that a controller outperforms our method.

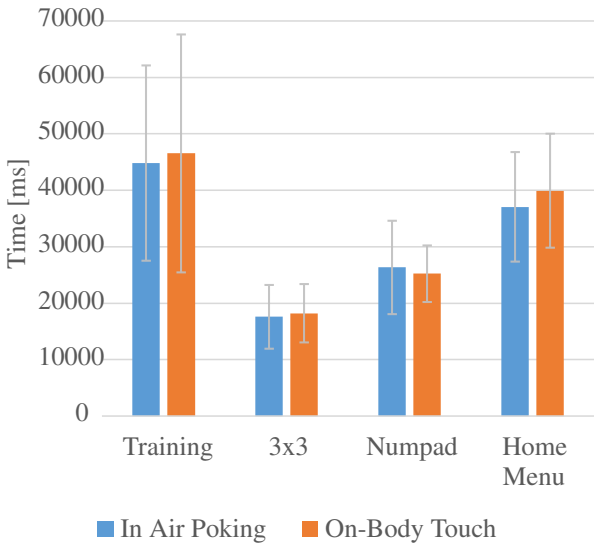

Figure 6. Average time of completion in milliseconds of different UI groups in Block 1 with confidence interval of 95%.

RESULTS

In this section, we report the completion times and errors of the different blocks and conditions. We also show the results of the subjective ratings. In Block 1 the condition In Air poking has a slightly smaller completion time, fewer errors and the participant preferred using this method compared to our method. In Block 2 we see that Controller has the best completion time, a small error count, and the best acceptance. Our method has the biggest average completion time but comes close to the time for the Pinching condition, our method has a small error count and comes closer to the acceptance of the controller than the Pinching condition.

Block 1

The average total times needed for completion are shown in Table 1. We excluded the Training group from the data. A complete overview with a 95% confidence interval is also in Figure 6. The average errors made during Block 1 can be found in Table 2.
Looking at the total completion times in Table 1, the condition In Air Poking was faster than the condition On-Body Touch by 2346 ms (considering 90 button activations). If we look at the UI element groups in detail, we see that In Air Poking was faster in group 3x3 (20 button activations) by 604 ms and in group Home Menu (30 button activations) by 2855 ms. Our

Condition	Total	3x3	Numpad	Home Menu
In Air Poking	81012	17597	26344	37070
On-Body Touch	83359	18202	25230	39926

Table 1. In this table we show the average completion time under a certain condition of Block 1 in milliseconds. The Total value under a condition is the sum of the other values regarding a UI element group.

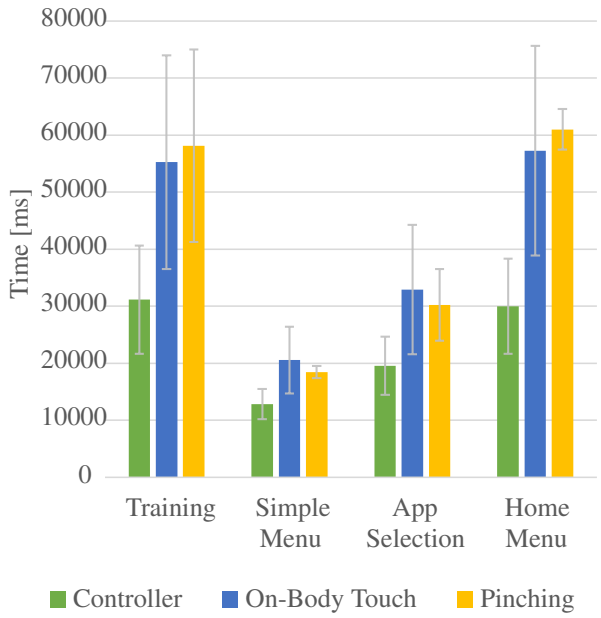

Figure 7. Average time of completion in milliseconds of different UI groups in Block 2 with confidence interval of 95%.

method was faster in Numpad (40 button activations) by 1113 ms. If we look at the average errors made during the study (considering 90 button activations), we see that In Air Poking with a total of 2.75 has fewer errors than our method with a total of 8.5. On average, In Air Poking has 0.75 errors less in the group 3x3 (20 button activations), 2 errors less in Numpad (30 button activations), and 3 errors less in Home Menu (40 button activations).

Noteworthy differences in the survey are that In Air Poking feels more natural (4.5 out of 5.0) than our method (3.75) and the liking to use the interaction technique is higher with In Air Poking (4.75) than with our method (4.0). Smaller differences were found in our method (5.0) being easier to solve than the other (4.75), In Air Poking (3.0) being better when it comes to the activation of the right button than our method (2.75) and that In Air Poking (1.5) does not feel as complex as our method (1.75). We found no difference regarding the perceived speed and fatigue.

Condition	Total	3x3	Numpad	Home Menu
In Air Poking	2.75	0.75	0.5	1.5
On-Body Touch	8.5	1.5	2.5	4.5

Table 2. In this table we show the average errors under a certain condition in Block 1. The Total value under a condition is the sum of the other values regarding a UI element group.

Block 2

The average total times needed for completion are shown in table 3. A complete overview with a 95% confidence interval is also in Figure 7. The average errors made during Block 2 can be found in Table 4.

The condition Controller with a total (90 panel activations) average completion time of 62354 ms was the fastest, followed by Pinching with 109662 ms and On-Body Touch with 110657 ms. Pinching has a smaller average completion time compared to our method in the groups Simple Menu (20 panel activations) and App Selection (30 panel activations), whereas our method has a smaller completion time in Home Menu (40 panel activations). The condition Controller has the smallest average completion time in every group. If we look at the average total (90 panel activations) errors made during the study, we see that Controller and our method have 4 as value and Pinching has 7. A notable difference is in the group Home Menu, where Controller has 1.3 errors on average, our method 2.3, and Pinching 4.

Condition	Total	Simple Menu	App Selection	Home Menu
Controller	62354	12835	19555	29963
On-Body Touch	110657	20539	32896	57222
Pinching	109662	18454	30221	60986

Table 3. In this table we show the average completion time under a certain condition of Block 2 in milliseconds. The Total value under a condition is the sum of the other values regarding a UI element group.

The survey shows that the Controller condition has the best evaluation except for activating the right UI element with a value of 3.0 out of 5.0 whereas for pinching it is 3.25 and for our method 3.5. Noteworthy is the gap regarding how fast one was able to solve the task. Controller got a 5.0 in comparison to our method with a value of 4.0 and Pinching with the lowest value of 2.25. The values for the Pinching method are lower than these of the other two except for the already discussed question regarding activating the right button. The different scores can be seen in detail in Figure 9.

Condition	Total	Simple Menu	App Selection	Home Menu
Controller	4	0.7	2	1.3
On-Body Touch	4	0.7	1	2.3
Pinching	7	1	2	4

Table 4. In this table we show the average errors under a certain condition of Block 2. The Total value under a condition is the sum of the other values regarding a UI element group.

DISCUSSION

The results of the study show that our method comes close to state-of-the-art interaction methods regarding completion time and user experience. It confirmed that a commercially available accelerometer can be used for on-body touch interaction.

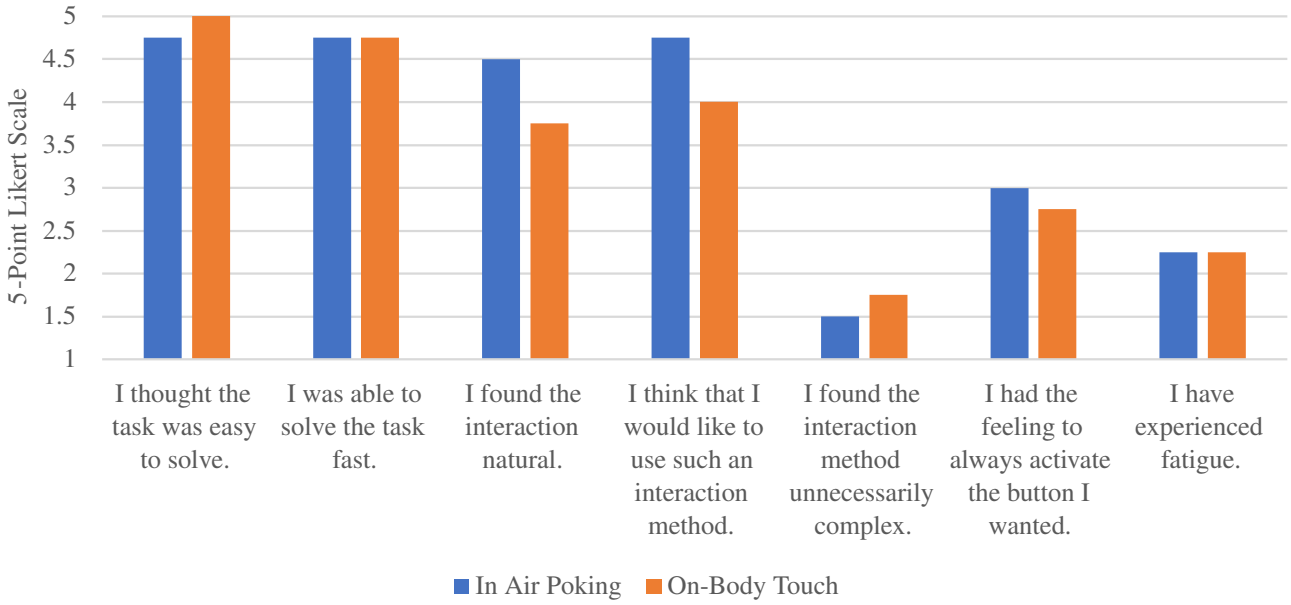

Figure 8. Evaluation of the questionnaire regarding block 1. The two conditions are shown in different colors and the values are the average over all answers.

We need to take into consideration that this is a pilot study with four participants. The number of participants is too low to have a significant result. As we saw in the graphs where the 95% confidence interval is too big for a comparison between the different conditions. Further studies with more participants are needed.
For the button selection with fingers (Block 1) our method was slightly slower with an average activation time per element of 926 ms to 900 ms for the In Air Poking. This difference is rather small and in practice negligible if the activation time is not the only focus. For one we do not activate this many buttons in a row and on the other hand, the participant did not report any difference in the completion time. Even though the questionnaire has similar results when it comes to the activation of the right buttons it does not get reflected in the error count. Our method has on average 8.5 errors whereas In Air Poking has only 2.75. The higher error count is due to our method being not that natural in comparison to In Air Poking that directly copies the interaction of a real button. With further use, our method most likely will feel more natural and decrease errors.

Looking at the different UI groups we see that the time it took the participant to complete 3x3 and Numpad was comparable to the other method or even faster. The buttons in these groups were close to each other and this means that the readjustments from the hand were not that big. Readjusting the hand can be a cause for a slower completion time with our method as we discuss in Limitations and Future work. Especially the completion of Numpad stood out where our method has a shorter average completion time. An advantage of our method here is that in VR it is enough to hover over the desired button and this is faster and easier than making a poking gesture for each one. The touch interaction in the real world slows our process a bit down.

A bigger difference can be seen with the group Home Menu. Even though there are fewer buttons involved than with Numpad the arrangement of the buttons differs. Now there are some rather small buttons spaced out that need a readjusting of the hand to get to the desired button and this takes up more time. The error count for both methods is the highest in this group.

The results with our method have a higher error count than In Air Poking. The activation of wrong buttons can also lead to some of the time differences seen before. Each time a participant is activating a wrong button the right button still needs to be triggered and the hand position must be readjusted. With our method, an error can also happen when the index finger hovers over a button and the participant unwanted triggered a tap event. Such a scenario happened sometimes during the study.

As expected, the use of a controller outperformed our method when it comes to interactions with rays (Block 2). The controller has also the best overall rating in the survey. When we compare the Pinching method with our own, we see that our method provides better or comparable results. The average activation time per element with pinching was 1218.5 ms whereas with our method 1229.5 ms. In comparison, Controller has an average activation time per element of 692.8 ms. Our method had an average error count of 4, the same as Controller, whereas Pinching has an error count of 7. In the survey, we see that the participants gave our method a better rating. The novelty effect could play a role here even though it is a minor factor, and it gets also diminished by the fact that for some participants the Pinching method was also new.

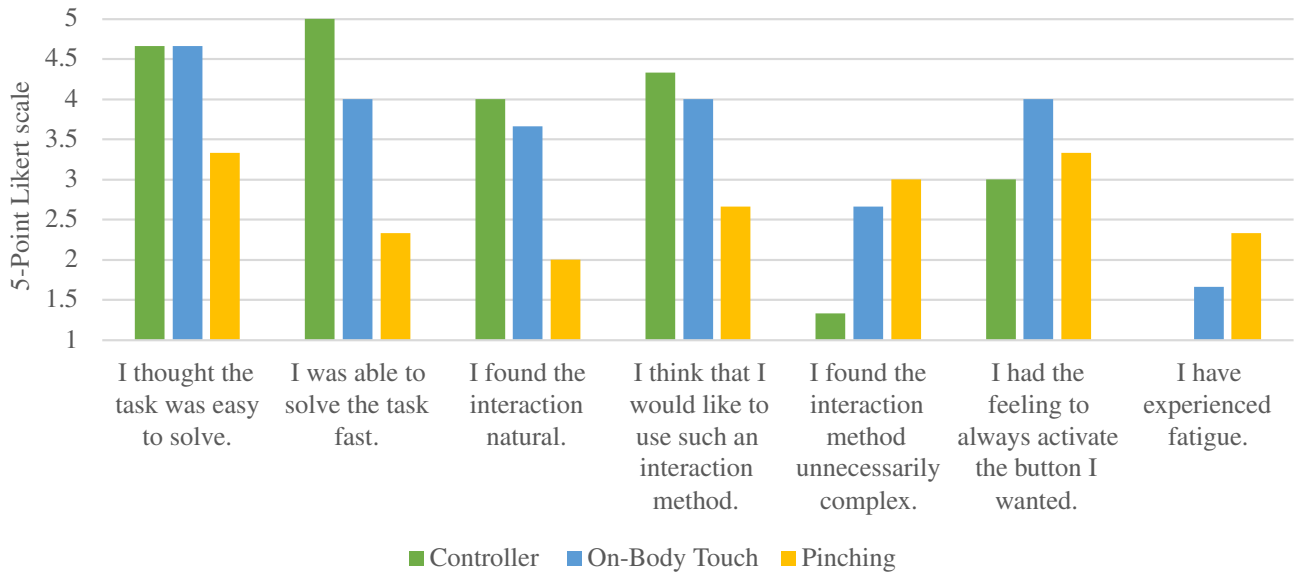

Figure 9. Evaluation of the questionnaire regarding block 2. The values are the average over all answers.

Especially in situations where a controller is not usable or the interaction of hands is desired such as augmented reality, our method is a viable option. The difference in the average activation time per element between our method and Pinching is very small. Small is also the error count for our method at least compared to the error count of Pinching. If we look at the UI groups in detail, we see that our method is faster in the more complex group Home Menu at least compared to Pinching. Whereas a bit slower with the other groups.

From the way how we cast the ray, our method is completely identical to Pinching. However, the activation of an element differs greatly. And even though the average activation time is comparable between the two methods, the total error count and evaluation of the questionnaire are not.

The evaluation of the survey showed a relatively low score for Pinching. Our study builds on the Oculus Framework and uses their method calls to see if the participant pinches. Compared to Oculus Home we do not provide any visual guidance regarding how much one should close and open the fingers to successfully pinch. This can lead to an unsatisfactory user experience. Additionally, the method suffers from occlusion. Here our method has an advantage since it always detects a touch. It is independent of the field of view of the HMD or occlusion of the own body parts.

Limitations and Future work

The study shown here is only a pilot and not significant enough for a good representation. Further studies will need more participants to see if there are significant differences between the methods.

Future studies based on our method should incorporate more participants, add some more tasks and experiment with different touch positions on the body. In this work, we focused only on menu item selection, but the method can be used for other interactions and applications too. Improving the limitations mentioned in this section can lead to better results in future studies.

One of the main limitations is that our method can only detect a tap. In combination with finger tracking, this one tap can be very versatile, however, it is not as functional as a controller. One improvement to make it more functional can be to detect which finger made a tap as seen in TapID, this may need some additional hardware. For this project, we decided to go with a backoff time after a tap and this creates a limit on how fast taps can be identified after each other. Our rather simple RCS method limits the accuracy of detecting a tap further. More involved methods will help to better distinguish taps from each other.

Another limitation is the way we let the participants tap. To make it uniform we let them tap the arm of the dominant hand with fingers of the non-dominant hand. Especially with the ray cast method, there is the problem that the arm of the dominant hand moves to point at the desired panel, and the non-dominant hand must move accordingly to touch the arm. This can lead to an unnatural process of moving the dominant hand to point at the desired object and then bringing the non-dominant hand in position for a tap. One solution for this is to let the non-dominant hand be stationary i.e., on the thigh.

Right now, we send the accelerometer data over a serial connection to the computer. This is not ideal for use in combination with VR. For our study, we did not investigate wireless connections but a wireless connection with a latency that is fast enough to send the data would solve this problem.

With the possibility of on-body touch and finger tracking, we can place the UI elements on the body and directly interact with them. An example is to place the UI elements on the non-dominant hand. This leads to an easy interaction for the

dominant hand. In this manner, we get also rid of the problem to readjust the hand with which we are touching since the place of the UI element in VR corresponds to the place where the touch event in the real world occurs. The haptic feedback from the touch agrees with the touch position. In this work, we did not investigate this further. Right now, there are still some tracking problems especially when the two hands interact with each other but in the future, this will also improve.

CONCLUSION

We have presented a method for on-body touch interaction that can be easily implemented with a commercially available IMU. The contact with the real body provides a form of haptic feedback. In cooperation with finger tracking, different ways of interactions in VR are possible. Our study showed that the accuracy of our method is comparable to current interaction technologies and accepted by the participants. The subjective ratings indicate that our method is preferred when compared to Pinching. Overall, we believe that our method is a viable option as an input method for hands-free menu item selection in VR.

REFERENCES

[1] Arduino . 2019a. Arduino Nano 33 IoT | Arduino Official Store. (2019). https://store.arduino.cc/arduino-nano-33-iot

[2] Facebook . 2019b. Getting Started with Hand Tracking on Oculus Quest 2 and Quest. (2019). https://support.oculus.com/articles/headsets-and-accessories/controllers-and-hand-tracking/hand-tracking-quest-2

[3] Facebook . 2019c. Oculus Device Specifications | Oculus Developers. (2019). https://developer.oculus.com/resources/oculus-device-specs/

[4] Facebook . 2020. Hand Tracking in Unity | Oculus Developers. (2020). https://developer.oculus.com/documentation/unity/unity-handtracking/

[5] Facebook . 2021a. Oculus. (2021). https://support.oculus.com/articles/headsets-and-accessories/controllers-and-hand-tracking/index-controllers-hand-tracking/

[6] Facebook . 2021b. Oculus Quest 2: Our Most Advanced All-in-One VR Headset | Oculus. (2021). https://www.oculus.com/quest-2/

[7] HTC . 2021c. Accessories for VIVE | VIVE European Union. (2021). https://www.vive.com/eu/accessory/

[8] Microsoft . 2021d. Microsoft HoloLens | Mixed Reality Technology for Business. (2021). https://www.microsoft.com/en-gb/hololens

[9] STMicroelectronics . 2017. LSM6DS3 iNEMO inertial module: always-on 3D accelerometer and 3D gyroscope Datasheet. (2017). https://www.st.com/resource/en/datasheet/lsm6ds3.pdf

[10] Ahmad Akl, Chen Feng, and Shahrokh Valaee. 2011. A Novel Accelerometer-Based Gesture Recognition System. *IEEE Transactions on Signal Processing* 59, 12 (2011), 6197–6205. DOI: http://dx.doi.org/10.1109/TSP.2011.2165707

[11] Brian Amento, Will Hill, and Loren Terveen. 2002. The Sound of One Hand: A Wrist-Mounted Bio-Acoustic Fingertip Gesture Interface. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems (CHI EA '02)*. Association for Computing Machinery, New York, NY, USA, 724–725. DOI: http://dx.doi.org/10.1145/506443.506566

[12] Niloofar Dezfuli, Mohammadreza Khalilbeigi, Jochen Huber, Florian Müller, and Max Mühlhäuser. 2012. PalmRC: Imaginary Palm-Based Remote Control for Eyes-Free Television Interaction. In *Proceedings of the 10th European Conference on Interactive TV and Video (EuroITV '12)*. Association for Computing Machinery, New York, NY, USA, 27–34. DOI: http://dx.doi.org/10.1145/2325616.2325623

[13] Yizheng Gu, Chun Yu, Zhipeng Li, Weiqi Li, Shuchang Xu, Xiaoying Wei, and Yuanchun Shi. 2019. Accurate and Low-Latency Sensing of Touch Contact on Any Surface with Finger-Worn IMU Sensor. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. Association for Computing Machinery, New York, NY, USA, 1059–1070. DOI: http://dx.doi.org/10.1145/3332165.3347947

[14] Sean Gustafson, Christian Holz, and Patrick Baudisch. 2011. Imaginary Phone: Learning Imaginary Interfaces by Transferring Spatial Memory from a Familiar Device. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. Association for Computing Machinery, New York, NY, USA, 283–292. DOI: http://dx.doi.org/10.1145/2047196.2047233

[15] Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. 2011. OmniTouch: Wearable Multitouch Interaction Everywhere. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. Association for Computing Machinery, New York, NY, USA, 441–450. DOI: http://dx.doi.org/10.1145/2047196.2047255

[16] Chris Harrison, Desney Tan, and Dan Morris. 2010. Skinput: Appropriating the Body as an Input Surface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. Association for Computing Machinery, New York, NY, USA, 453–462. DOI: http://dx.doi.org/10.1145/1753326.1753394

[17] Christian Holz, Tovi Grossman, George Fitzmaurice, and Anne Agur. 2012. Implanted User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. Association for Computing Machinery, New York, NY, USA, 503–512. DOI: http://dx.doi.org/10.1145/2207676.2207745

[18] Hsin-Liu (Cindy) Kao, Christian Holz, Asta Roseway, Andres Calvo, and Chris Schmandt. 2016. DuoSkin: Rapidly Prototyping on-Skin User Interfaces Using Skin-Friendly Materials. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers (ISWC '16)*. Association for Computing Machinery, New York, NY, USA, 16–23. DOI: http://dx.doi.org/10.1145/2971763.2971777

[19] Jarrod Knibbe, Diego Martinez Plasencia, Christopher Bainbridge, Chee-Kin Chan, Jiawei Wu, Thomas Cable, Hassan Munir, and David Coyle. 2014. Extending Interaction for Smart Watches: Enabling Bimanual around Device Control. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems (CHI EA '14)*. Association for Computing Machinery, New York, NY, USA, 1891–1896. DOI: http://dx.doi.org/10.1145/2559206.2581315

[20] Gierad Laput, Robert Xiao, Xiang 'Anthony' Chen, Scott E. Hudson, and Chris Harrison. 2014. Skin Buttons: Cheap, Small, Low-Powered and Clickable Fixed-Icon Laser Projectors. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. Association for Computing Machinery, New York, NY, USA, 389–394. DOI: http://dx.doi.org/10.1145/2642918.2647356

[21] Gierad Laput, Robert Xiao, and Chris Harrison. 2016. ViBand: High-Fidelity Bio-Acoustic Sensing Using Commodity Smartwatch Accelerometers. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. Association for Computing Machinery, New York, NY, USA, 321–333. DOI: http://dx.doi.org/10.1145/2984511.2984582

[22] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. 2009. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing* 5, 6 (2009), 657–675. DOI:http://dx.doi.org/10.1109/PERCOM.2009.4912759 PerCom 2009.

[23] Joanne Lo, Doris Jung Lin Lee, Nathan Wong, David Bui, and Eric Paulos. 2016. Skintillates: Designing and Creating Epidermal Interactions. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems (DIS '16)*. Association for Computing Machinery, New York, NY, USA, 853–864. DOI: http://dx.doi.org/10.1145/2901790.2901885

[24] Manuel Meier, Paul Streli, Andreas Fender, and Christian Holz. 2021. TaplD: Rapid Touch Interaction in Virtual Reality using Wearable Sensing. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. IEEE, Piscataway, NJ, 519–528. DOI: http://dx.doi.org/10.1109/VR50410.2021.00076

[25] Aditya Shekhar Nittala, Anusha Withana, Narjes Pourjafarian, and Jürgen Steimle. 2018. Multi-Touch Skin: A Thin and Flexible Multi-Touch Sensor for On-Skin Input. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. DOI: http://dx.doi.org/10.1145/3173574.3173607

[26] Masa Ogata and Michita Imai. 2015. SkinWatch: Skin Gesture Interaction for Smart Watch. In *Proceedings of the 6th Augmented Human International Conference (AH '15)*. Association for Computing Machinery, New York, NY, USA, 21–24. DOI: http://dx.doi.org/10.1145/2735711.2735830

[27] Daniel Schneider, Alexander Otte, Axel Simon Kublin, Alexander Martschenko, Per Ola Kristensson, Eyal Ofek, Michel Pahud, and Jens Grubert. 2020. Accuracy of Commodity Finger Tracking Systems for Virtual Reality Head-Mounted Displays. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. IEEE, Atlanta, GA, USA, 804–805. DOI:http://dx.doi.org/10.1109/VRW50115.2020.00253

[28] Unity Technologies. 2019. Unity - Unity. (2019). https://unity.com/

[29] Martin Weigel, Tong Lu, Gilles Bailly, Antti Oulasvirta, Carmel Majidi, and Jürgen Steimle. 2015. ISkin: Flexible, Stretchable and Visually Customizable On-Body Touch Sensors for Mobile Computing. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. Association for Computing Machinery, New York, NY, USA, 2991–3000. DOI: http://dx.doi.org/10.1145/2702123.2702391

[30] Yang Zhang, Wolf Kienzle, Yanjun Ma, Shiu S. Ng, Hrvoje Benko, and Chris Harrison. 2019. ActiTouch: Robust Touch Detection for On-Skin AR/VR Interfaces. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. Association for Computing Machinery, New York, NY, USA, 1151–1159. DOI: http://dx.doi.org/10.1145/3332165.3347869