Multimodal Technologies and Interaction

MDPI

Article

Observations and Considerations for Implementing Vibration Signals as an Input Technique for Mobile Devices

Thomas Hrast, David Ahlström and Martin Hitz *

Interactive Systems Research Group, Department of Informatics Systems, University of Klagenfurt, Universitätsstraße 65–67, 9020 Klagenfurt am Wörthersee, Austria; thhrast@edu.aau.at (T.H.); david.ahlstroem@aau.at (D.A.)
* Correspondence: martin.hitz@aau.at; Tel.: +43-463-2700-3512

Abstract: This work examines swipe-based interactions on smart devices, like smartphones and smartwatches, that detect vibration signals through defined swipe surfaces. We investigate how these devices, held in users' hands or worn on their wrists, process vibration signals from swipe interactions and ambient noise using a support vector machine (SVM). The work details the signal processing workflow involving filters, sliding windows, feature vectors, SVM kernels, and ambient noise management. It includes how we separate the vibration signal from a potential swipe surface and ambient noise. We explore both software and human factors influencing the signals: the former includes the computational techniques mentioned, while the latter encompasses swipe orientation, contact, and movement. Our findings show that the SVM classifies swipe surface signals with an accuracy of 69.61% when both devices are used, 97.59% with only the smartphone, and 99.79% with only the smartwatch. However, the classification accuracy drops to about 50% in field user studies simulating real-world conditions such as phone calls, typing, walking, and other undirected movements throughout the day. The decline in performance under these conditions suggests challenges in ambient noise discrimination, which this work discusses, along with potential strategies for improvement in future research.

Keywords: mobile device; smartphone; smartwatch; vibration signal; design factors; support vector machine (SVM); feature vector

Citation: Hrast, T.; Ahlström, D.; Hitz, M. Observations and Considerations for Implementing Vibration Signals as an Input Technique for Mobile Devices. *Multimodal Technol. Interact.* **2024**, *8*, 76. https://doi.org/10.3390/mti8090076

Academic Editors: Wei Liu, Jan Auernhammer, Takumi Ohashi, Di Zhu and Kuo-Hsiang Chen

Received: 19 May 2024
Revised: 17 June 2024
Accepted: 7 August 2024
Published: 2 September 2024

1. Introduction

Today, interactions on screens are common on mobile devices such as smartphones and smartwatches. Kratz et al. [1] observed a paradigm shift in mobile device research, transitioning from on-screen interactions to more advanced forms of interaction. This transition is facilitated by the integration of accelerometers, gyroscopes, and depth-camera technologies in devices like smartphones and smartwatches, enabling novel interaction possibilities beyond traditional hand–eye interactions [1].

An interaction challenge on mobile devices arises in scenarios such as cooking, where touchscreen interaction is hindered by wet hands. Another example pertains to museum visits, where users can gain additional information by swiping their fingers over surfaces, such as a conch shell, to obtain insights about the object and its associated sea life Han et al. [2]. While accessibility tools like screen readers, Braille terminals, and talking browsers are effective for text-based content, they have limitations when it comes to handling visual content [3]. Moreover, Harrison et al. [4] discussed the use of Braille as a human-readable tactile encoding that can be interpreted by mobile devices. Swiping the index finger across the Braille can display the corresponding content on the screen of the mobile device. Therefore, we want to take a closer look at such swipe gestures. See Appendix A for a discussion of the term *swipe gesture*.

Swipe gestures are performed by users while holding their smartphones and swiping across different swipe surfaces. Swipe gestures are used to execute an input command

on mobile devices [2,4,5]. Swipe surfaces can be grooves, notches, natural textures, or artificially created textures [2,4].

The underlying concept is analogous to a record player and a vinyl long play (LP). The LP contains the encoded information and the needle picks up the information as the LP rotates on the turntable. Therefore, Harrison et al. [4] and Han et al. [2] propose solid surfaces with different bump patterns for capturing these vibration signals. During a swipe gesture, the mobile device experiences slight vibrations. The built-in accelerometer in the device detects these minute vibrations and translates them into digital signals, which serve as input commands for the mobile device. The built-in accelerometer of mobile devices generates samples. These samples are used for calculations. These calculations, along with or without the raw samples, can be organized into a vector at regular intervals. Such a vector is commonly referred to as a feature vector [6–9], vector [10], or features [2,11–14]. We use in this work the term *feature vector*, represented as $\vec{Y}$, composed of $l_{\vec{Y}}$ feature elements $\left\{Y_1, Y_2, \ldots, Y_{l_{\vec{Y}}}\right\}$. Then, the feature vector $\vec{Y}$ is sent to a support vector machine (SVM) that classifies the different swipe surfaces. An SVM is a widely used machine learning (ML) algorithm that learns from examples to infer desired input-output behavior [15] without requiring explicit programming. By classifying the feature vector $\vec{Y}$, the SVM determines the corresponding content to be displayed on the mobile device, such as sending a message or performing touchscreen interactions. In this work, we want to expand on the findings of Harrison et al. [4] and Han et al. [2] by utilizing SVM while also other ML approaches, such as neural networks, would be possible for this work.

In this section, we explained how to use gestures as input on mobile devices using different swipe surfaces. We described swipe surface detection on mobile devices using SVM while users are wearing them and performing a swipe gesture over these solid surfaces with bump patterns. With these insights, we detail the research questions in the next section.

2. Research Question

Based on the ideas proposed by Harrison et al. [4] and Han et al. [2], we formulate the following research question for this work:

> «What factors affect the efficiency of vibration-based input methods on mobile devices when users swipe with their index finger under different swipe movement behaviors over textured surfaces while wearing the devices?»

In this work, we investigate the behavior of the feature vector $\vec{Y}$ when using SVMs to detect swipe surfaces. The factors are divided into human-determined and software-determined aspects [2]. The human-determined aspects relate to how users' swipes access different swipe surfaces. The software-determined aspects relate to how the feature vector is constructed, how the SVM is trained, and how it can classify the different swipe surfaces.

Unlike Han et al. [2], who used a *Bosch BNO055* vibration sensor, we used the built-in accelerometers in smartphones and smartwatches. Alternatively, an acoustic sensor can be employed to detect vibration signals. However, it is worth noting that acoustic sensors pose certain limitations, as the vibration signal and ambient noise often occupy the same frequency range [16]. Consequently, acoustic sensors may not be the optimal choice for environments with high levels of ambient noise [2,14,17–19], such as instances where a radio is playing [18].

The SVM classifies the sequentially streamed data from the built-in accelerometer in mobile devices. Porzi et al. [20] defined a swipe gesture as a time series of acceleration measurements. However, the data stream from the accelerometer does not have a defined ending. We represent it as $w = \{s_1, s_2, \ldots, s_\infty\}$. Each sample in this data stream is associated with a correlated time stamp, denoted as $\{t_1, t_2, \ldots, t_\infty\}$. The infinity data stream is indicated by the sample s_∞ with the linked timestamp t_∞. We take a specific subset of this time series, denoted as $w = \{S_1, S_2, \ldots, S_l\}$, where l represents the number of samples in this defined subset, and the corresponding time stamps are $\{t_1, t_2, \ldots, t_l\}$. From this subset w, we extract the feature vector $\vec{Y}$. The subset w can contain not only samples

from the swipe surfaces but also ambient noise. The presence of ambient noise can have a detrimental effect on the classification performance of SVMs [21]. Therefore, it is necessary to use appropriate features for the feature vector that will allow the SVM to effectively filter out the ambient noise while identifying the swipe surfaces.

The SVM must be trained before it can use a feature vector to classify different vibration signals. In the training phase, the SVM needs a label and a feature vector. The label is the name of the swipe surfaces. The feature vector $\vec{Y}$ describes the swipe surfaces. In addition, Spolaôr et al. [22] distinguish between multi-label and single-label learning. In multi-label learning, a feature vector $\vec{Y}$ is associated with multiple labels at the same time. In contrast, single-label learning only associates each feature vector with one swipe surface label. In this work, we focus on the single-label learning.

Among machine learning methods, supervised learning is the most commonly used one [15]. Two other learning methods are unsupervised learning and reinforcement learning [15]. Unsupervised learning involves the analysis of unlabeled data based on assumptions about the structural properties of the data, such as algebraic, combinatorial, or probabilistic [15]. In reinforcement learning, the information available in the training data is intermediate between supervised and unsupervised learning [15].

Finally, we discuss how gestures can be recognized with the accelerometer. Porzi et al. [20] distinguish between user-dependent and user-independent gestures recognition. In user-dependent gesture recognition, a user performs a few gestures before utilizing the recognition system, while user-independent recognition accounts for variations in the same gestures from different users without requiring a preliminary recording phase. User-independent recognition typically relies on a pretrained SVM [20]. The classification of different swipe surfaces will be user-independent. Users can use this interface without any additional configuration or settings.

In this section, we explore the research question from both human-determined and software-determined aspects. We outlined methods to train the SVM. In this work, we focus on a supervised method to train the SVM with single-label swipe surfaces and implement user-independent gesture recognition.

The remaining structure of the work is as follows: In Section 3, we discuss related work on human-determined and software-determined aspects. In Section 4, we conduct a user study to record and transform vibration signals into feature vectors. Then, in Section 5, we train the SVM for both smartphone and smartwatch setups with best and worst classification accuracy. In Section 6, we evaluate these setups with additional data and discuss the results with the literature in Section 7. Finally, Section 9 summarizes the findings and suggests further research.

In the next section, we review the existing literature on how mobile devices can recognize different swipe surfaces while performing a swipe gesture is performed over such swipe surfaces.

3. Related Work

The discussion of the literature focuses on human-determined and software-determined aspects. Open questions in the literature are the starting point for the laboratory user study in Section 4.

3.1. Human-Determined Aspects

Liu et al. [23] discovered that the behavior of a vibration signal closely resembles that of a wireless signal. As the vibration signal travels through one's hand, it experiences attenuation along the propagation path and exhibits reflection or diffraction when it encounters the boundary between different media, such as the skin and bones [23]. Various behavioral and physiological characteristics, such as touching force and contacting area, influence the propagation of the vibration signal [23]. However, the significance of the mobile device's location is not the sole consideration. The specific gesture articulated while swiping across the surface also plays a crucial role. Han et al. [2] observed that different swipe directions

yield varying outcomes, and the manner in which users swipe, such as using the nail or only the finger, also has an influence [2]. How users hold or wear mobile devices, as well as the appropriate swipes across different surfaces, remain uncertain factors in achieving a high classification accuracy of the surface using the SVM.

Another human-determined aspect is the location of the device used to detect vibration signals. Kefer et al. [24] found no significant difference in vibration detection between wrist-worn devices and vibration sensors positioned below the elbow. Han et al. [2] placed the vibration sensor on both the finger and the wrist, revealing minor differences. In the study conducted by Chen et al. [25], a vibration sensor was mounted on the wrist while participants tapped with their finger on the opisthenar. Additionally, Kim et al. [26] highlighted the rapid attenuation of vibrations within the body, limiting the detectable range. Furthermore, Ikeda et al. [27] established a relationship between the ankle's joint angle and the propagation amplitude, indicating that changes in the mechanical properties of the human body can impact frequency characteristics. Consequently, the feasibility of detecting vibration signals using smartphones and smartwatches remains uncertain.

Lukowicz et al. [17] highlighted the challenge of precisely defining the start and end points of a gesture. While previous work [2,24,28–30] focused on using accelerometer data to train SVMs, they did not specifically address the extraction of vibration signals from swipe gestures in a continuous time series. Even when considering a finite subset of this time series, they did not distinguish between swipe gestures and ambient noise occurring before and after the swipe gesture on the textured surface. Therefore, it remains unclear how to extract swipe gestures on textured surfaces from ambient noise that does not represent such gestures. Also, there is a lack of established methods for training SVMs to distinguish swipe gestures from ambient noise that does not represent such gestures.

Now, we transition from aspects determined by humans to those specified by software.

3.2. Software-Determined Aspects

As mentioned in Section 2, a subset w must be extracted from an infinite data stream $\vec{s}$ to allow the SVM is able to classify different textured surfaces. One possible approach is to use a sliding window technique to extract subsets of an $\vec{s}$ [6,25,29–32], which is explored next.

3.2.1. Sliding Window

The process of moving the window along the time series allows us to capture subsets of the data to construct the feature vector $\vec{Y}$ with a specified overlap. Some studies suggest a half overlap with the samples from the previous window [6,29,32,33]. However, Chen et al. [25] used a sliding window size of 35 ms with a 5 ms overlap. One concern raised by Han et al. [2] was that the accelerometer in mobile devices has a lower sampling rate compared with acoustic sensors, resulting in less sensitivity in detecting vibration signals. However, Lukowicz et al. [17] used a sampling rate of 100 Hz and successfully extracted vibration signal patterns. Owusu et al. [34] also investigated various sampling rates and found that 100 Hz is sufficient for detecting push vibration patterns with smartphones. This sampling rate is commonly used in mobile devices [2,13]. DeVaul and Dunn [35] used a sampling rate of 47 Hz to detect body motion using a microcontroller-based accelerometer sensor wearable.

If the sliding window is too large but cannot be filled with samples because the swipe gesture has already been completed, the Fast Fourier Transformation (FFT) cannot be computed. On the other hand, if the sliding window is extremely small, the entire window may be received, but it may not be sufficient to classify different swipe surfaces. Therefore, we outline the possible number of samples in a sliding window for a sample rate of about 100 Hz for classifying different swipe surfaces. Previous works have used different numbers of samples in a window, such as 10 samples [20], using a smartwatch with a low sample rate, 64 samples [2], 128 samples [36], 256 samples [33], and even 4096 samples [4], but using an acoustic sensor. However, when dealing with a low and limited sample rate, there may be occasional loss of swipe gesture information [33]. There-

fore, determining the appropriate number of samples to include in a sliding window for accurate classification of different swipe surfaces at a sample rate of approximately 100 Hz remains an open question.

3.2.2. Filter

Liu et al. [33] explain that random hand movements introduce high-frequency noise, motivating the use of filters. A filter is an algorithm used to select or modify certain samples from a vibration signal recorded by the accelerometer of a mobile device. It mathematically processes the signal in either the time or frequency domain. The goal of filters is to help identify swipe gestures on different surfaces while reducing the effect of ambient noise. Therefore, we discuss what type of filter should be implemented and what cut-off frequency ω_c should be used.

Han et al. [2] applied a first-order Butterworth high-pass filter with a cut-off frequency of 0.1 Hz to mitigate sensor drift and reduce accumulated errors. Chen et al. [25] utilized a Butterworth high-pass filter with a cut-off frequency of 20 Hz to eliminate ambient noise caused by human motion. In addition, they employed a Butterworth low-pass filter with a cut-off frequency of 300 Hz to address a tapped vibration signal estimated at 200 Hz. Similarly, Lukowicz et al. [17] employed a low-pass filter with a cut-off frequency of 50 Hz to filter out frequencies beyond this threshold. Srivastava et al. [37] used a low-pass Butterworth filter without specifying the cut-off frequency to remove ambient noise and enhance the detection of abnormal heart frequencies. DeVaul and Dunn [35], McGrath and Li [28], and Yurttadur and Karaçay [29] did not provide information about the cut-off frequencies. The effect of a Butterworth high-pass filter with an appropriate cut-off frequency ω_c to distinguish between the swipe gesture and ambient noise remains unclear.

3.2.3. Feature Vector

Selecting an appropriate feature vector $\vec{Y}$ is crucial for achieving high classification accuracy for different swipe surfaces [29,38]. However, the selection of elements for the feature vector remains an active research topic in supervised learning [22]. Saeys et al. [39] explain the motivation for choosing the right feature elements for $\vec{Y}$: (a) avoiding overfitting and improving the model's performance in supervised classification and clustering tasks, (b) creating faster and more cost-effective models, and (c) gaining deeper insights into the underlying data-generating processes. Overfitting occurs when an SVM model describes ambient noise in the data rather than the textured surface, leading to excellent performance on observed data but poor performance on unseen data [40]. A cost-effective model can provide accurate results of textured surfaces even with a small dataset to train the SVM and relatively low computational resources.

Now, take a closer look at the sliding window $w = \{S_1, \ldots, S_i, \ldots, S_l\}$ to define its properties. This sliding window is a subset of the infinity data stream $\{s_1, s_2, \ldots, s_\infty\}$ from the accelerometer. The ith sample S_i within w contains the acceleration sensor readings in three different directions x, y, and z. Therefore, we can represent S_i as $\left\{S_i^{(x)}, S_i^{(y)}, S_i^{(z)}\right\}^T$ [20]. We can interpret w as a matrix with dimensions $3 \times l$. Here, 3 represents the three acceleration directions, and l denotes the length of the sliding window. Equation (1) [6,20,32] illustrates the structure of this matrix.

$$w = \begin{pmatrix} S_{1,1}^{(x)} & S_{1,2}^{(x)} & S_{1,3}^{(x)} & \cdots & \cdots & \cdots & S_{1,l}^{(x)} \\ S_{2,1}^{(y)} & S_{2,2}^{(y)} & S_{2,3}^{(y)} & \cdots & S_{i,j}^{(\cdot)} & \cdots & S_{2,l}^{(y)} \\ S_{3,1}^{(z)} & S_{3,2}^{(z)} & S_{3,3}^{(z)} & \cdots & \cdots & \cdots & S_{3,l}^{(z)} \end{pmatrix} \tag{1}$$

From Equation (1), we extract the feature vector $\vec{Y}$. Inspired by literature [2,24,28,29,32,41], Equations (2)–(4) represent a general formulation for possible feature elements of the feature vector.

$$\vec{Y} = \{\Xi_1, \ldots, \Xi_n\} \tag{2}$$

$$\vec{Y} = \left\{\Xi_1, \ldots, \Xi_n, S_{1,1}^{(x)}, \ldots, S_{1,l}^{(x)}, S_{2,1}^{(y)}, \ldots, S_{2,l}^{(y)}, S_{3,1}^{(z)}, \ldots, S_{3,l}^{(z)}\right\} \tag{3}$$

$$\vec{Y} = \left\{\hat{\Xi}_1, \ldots, \hat{\Xi}_n, \hat{S}_{1,1}^{(x)}, \ldots, \hat{S}_{1,l}^{(x)}, \hat{S}_{2,1}^{(y)}, \ldots, \hat{S}_{2,l}^{(y)}, \hat{S}_{3,1}^{(z)}, \ldots, \hat{S}_{3,l}^{(z)}\right\} \tag{4}$$

The index l in Equations (2)–(4) corresponds to the length of w in Equation (1). Nevertheless, the length l of the sliding window w is not necessarily equal to the length $l_{\vec{Y}}$ of the feature vector $\vec{Y}$. For instance, if the sliding window w contains $l = 64$ samples, the resulting length $l_{\vec{Y}}$ of $\vec{Y}$ will comprise 3 acceleration directions $\cdot$ 64 samples $= 192$ elements when considering only accelerometer samples for $\vec{Y}$.

The feature vector $\vec{Y}$ can have feature elements in time, frequency, or a combination of both. To transform the matrix in Equation (1) from the time domain to the frequency domain, we use the Fast Fourier Transform (FFT). The symbol Ξ represents elements in the time domain, while $\hat{\Xi}$ corresponds to the frequency domain. The symbol S represents raw samples recorded by the accelerometer, which are in the time domain, while $\hat{S}$ represents the converted samples in the frequency domain.

The symbol Ξ, for instance in Equation (2), is a general form that describes the vibration signal, including calculations from statistics, physics, or human behavior. We can have n of Ξ elements in $\vec{Y}$.

Vibration signal descriptions based on statistics include metrics such as the minimum [6,24,34,41], maximum [24,34,41], mean [2,24,28,34], standard deviation [2,6,24,28,41], skewness [2,28], and kurtosis [2,28,29]. Han et al. [2] considered finger swiping directions, forearm pointing direction, and finger pointing direction as part of the vibration signal descriptions, which are based on human behaviors.

Physics-based descriptions involve concepts such as power spectral density (PSD) [24,32,41,42], mel-frequency cepstral coefficients (MFCC) [23,25,37,38,43], crest factor [29], or the presence of two negative peaks within one window [30] to describe the vibration signal of a textured swipe surface.

Previously, the vibration signal was described separately for each acceleration direction. However, Miluzzo et al. [32] and McGrath and Li [28] proposed incorporating correlation characteristics between acceleration directions obtained from the accelerometer. To facilitate this calculation, the sliding window is considered.

Equation (1) illustrates the matrix structure of w, which can be used, for instance, to determine the norm $\|w\|$. Miluzzo et al. [32] and McGrath and Li [28] introduced the 1-norm, Infinity norm, and Frobenius norm as measures to describe the vibration signal of the swipe gesture over textured surfaces.

Various other combinations are also possible, such as combining the vibration signal description in the frequency domain $\hat{\Xi}$ with samples from the time domain S.

With the structure of the feature vector defined, our discussion now focuses on determining the optimal number of elements within the feature vector $\vec{Y}$. Additionally, we discuss the process of selecting the elements for $\vec{Y}$.

Zhang et al. [7] emphasized the importance of having a feature vector with a small number of highly significant elements. John et al. [11] noted that removing certain samples from $\vec{Y}$ leads to a decrease in the probability of correctly classifying the textured surface, indicating their higher significance. Miluzzo et al. [32] reported 273 elements in $\vec{Y}$, while Owusu et al. [34] described 46 Ξs, and Kefer et al. [24] reported over 50 different Ξs. Mehrnezhad et al. [41] applied 36 Ξs in the time and frequency domains, resulting in a total of 72 descriptive elements for textured swipe surfaces. However, Liu et al. [31] and Funato and Takemura [44] suggested that the power spectral density (PSD) alone is sufficient to achieve high classification accuracy. It is important to note that the methods mentioned for describing the vibration signal were primarily used for push vibrations [25,28,31,42] or stepping patterns [29,30], which are closely related to push patterns.

Han et al. [2] used 109 elements for $\vec{Y}$, including 13 $\hat{\Xi}$ elements and 96 $\hat{S}$ samples. Comparing this feature vector length to that of Miluzzo et al. [32], Han et al.'s $\vec{Y}$ [2] is

approximately 60% smaller. The ratio is calculated as $109/273 \approx 0.4$. Subtracting this ratio from 1 gives 0.6 or 60%. This suggests that it may be possible to reduce the feature vector while maintaining a similar level of accuracy.

Cho et al. [13] used 24 elements for their feature vector, including vibration signal descriptions in both the time and frequency domains. A potential advantage of reducing the size of $\vec{Y}$ is the lower computational effort required, especially when dealing with infinite time series. Parera et al. [6] found that having seven elements in $\vec{Y}$ resulted in the highest classification accuracy within the range of zero to 30 elements. Beyond seven elements, the correct classification rate decreased. However, it is currently unclear how many elements are required in the feature vector to achieve high classification accuracy. Additionally, Önem [45] suggested the need to identify the most relevant samples in $\vec{Y}$. Based on these considerations, it might be possible to use a smaller number of elements in the feature vector by selecting the most informative ones. Önem [45] mentioned that an excessively large feature vector is impractical, and in practical supervised learning algorithms, the aim is to find a satisfactory $\vec{Y}$ rather than an optimal one. Another argument for reducing the number of elements in the feature vector is the computational effort required for each element. Parera et al. [6] found that computing the standard deviation involved significant processing time, but even without it, an acceptable classification rate was achieved Parera et al. [6]. Therefore, the time and computational effort associated with each selected feature vector element should also be taken into account. If the calculation of an element takes too long, it may pose challenges in classifying each received w in a timely manner.

Finally, we consider the SVM kernel, which is also part of the software-determined aspect.

3.2.4. SVM Kernel

SVMs are well-known algorithms that utilize kernel substitution [46]. Selecting an appropriate kernel and optimizing kernel parameters can lead to improved classification rates for surfaces [38,47]. To develop a software application that uses SVM to classify different textured surfaces with swipe gestures, the Java library *libSVM* [10] provides options to configure different kernels, and we explore the use of such different kernels below.

The most common kernels are linear, polynomial, sigmoid, and radial basis function (RBF) [48,49]. The RBF kernel is also known as the Gaussian kernel [48,49]. Parera et al. [6] successfully classified daily activity vibrations using the RBF kernel. Qin et al. [50] employed acoustic sensors to detect and classify keystroke vibration signals, utilizing the RBF kernel to handle ambient noise. McGrath and Li [28] and Liu et al. [14] applied linear kernels for classifying push vibrations, while Han et al. [2] used a linear kernel to classify swiping vibration patterns. Miluzzo et al. [32] used both linear and RBF kernels to classify push vibrations. Cho et al. [13] employed the RBF kernel to detect different surfaces using the built-in vibrator and accelerometer of a smartphone. It is worth noting that in their approach, the smartphone emitted the vibration signals rather than measuring them. The default kernel in libSVM is RBF [10]. However, it remains unclear if a specific kernel leads to high classification accuracy for different textured surfaces.

To conclude this section, we have discussed both human-determined and software-determined factors for classifying textured surfaces. Human-determined aspects include how users perform swipe gestures on these surfaces. Software-determined factors include creating a sliding window, applying a filter, generating the feature vector, and selecting the SVM kernel. However, the existing literature lacks clear guidelines on how users should execute these gestures, leading to uncertainties in achieving high classification accuracy. To address this gap, we conducted a laboratory user study to better understand the requirements for accurately classifying textured surfaces during swipe gestures. Details of this study are provided in the following section.

4. Laboratory User Study

We explain the swipe surfaces, the human-determined aspects, and the software-determined aspects in this laboratory user study that was conducted.

4.1. Swipe Surfaces

We present a total of five different swipe surfaces. Two of the surfaces are intended for on-body interfaces [51], while the other three are solid and separate from the human body.

According to Harrison et al. [51], on-body interfaces offer novel interactive possibilities, transforming our hands into input/output devices. The rationale behind on-body systems is as follows: (a) they are socially more acceptable than speech interfaces and ergonomically superior to gestural interfaces; (b) the skin provides a larger interaction area compared with mobile devices; (c) tactile feedback is delivered to users through their own body; (d) they tap into muscle memory, hand-eye coordination, and familiarity with one's own body [52]; and (e) they facilitate eyes-free interaction [51,53].

The selection of the solid surfaces has been influenced, for example, by Han et al. [2], the Scratch Input technique [54], Skinput [53], and Touché [55].

Figure 1a illustrates the considered on-body and solid surfaces.

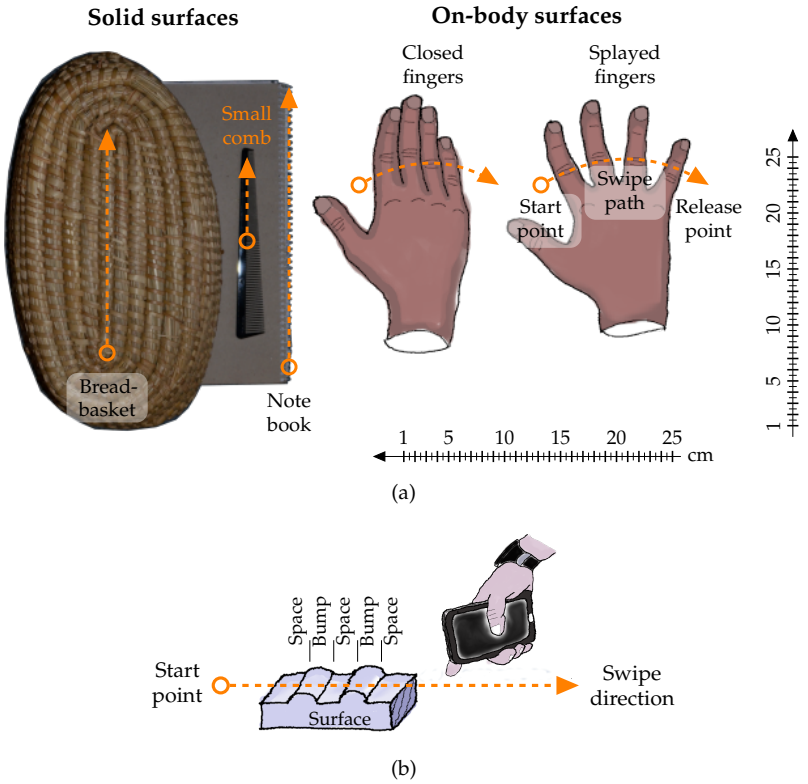
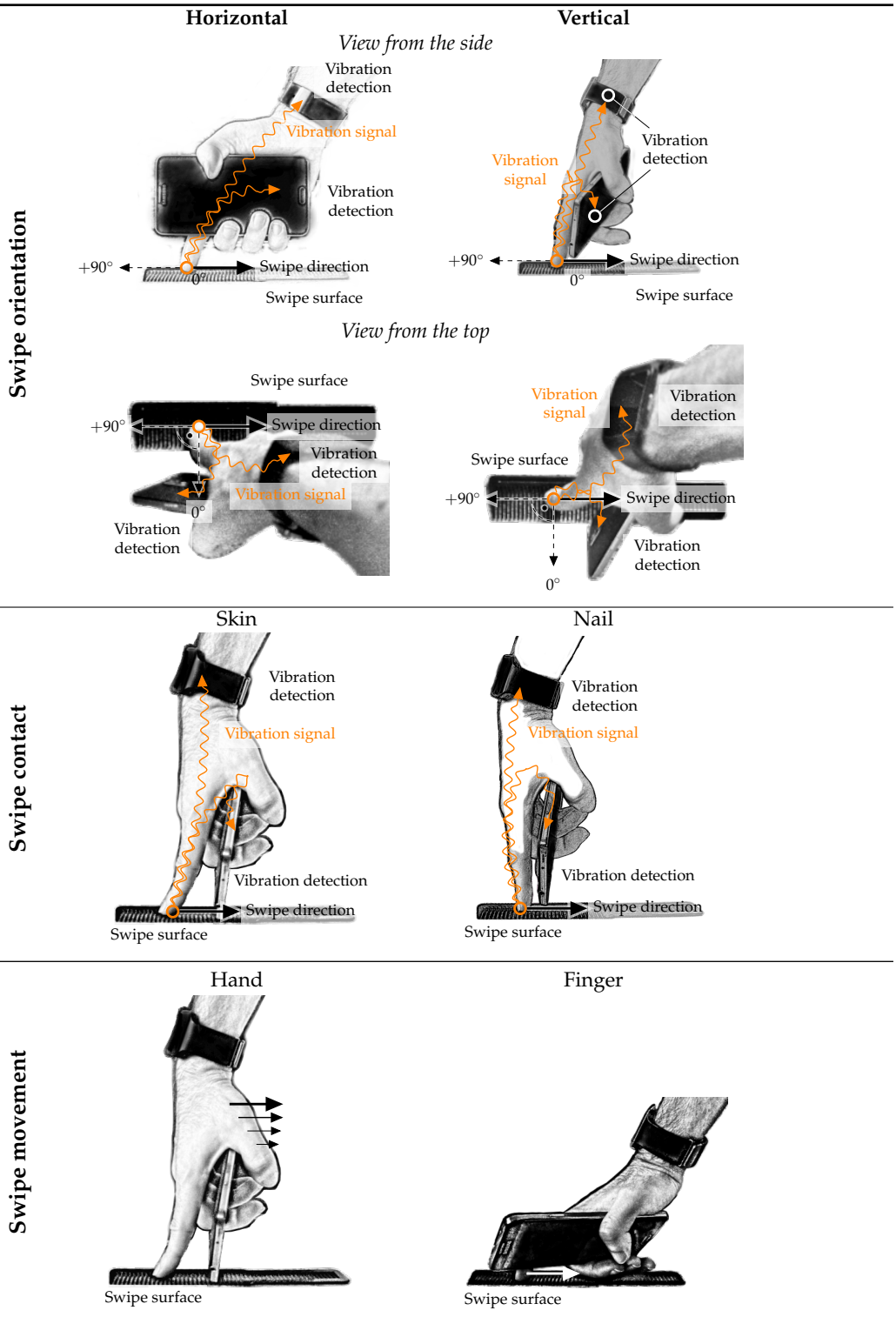


Figure 1. Picture (**a**) shows the swipe path on the different surfaces in this work, whereas picture (**b**) points out the relationship between bumps and spaces on these surfaces.

The key factor in our laboratory user studies is the relationship between the bumps and the spaces on the swipe surfaces [2]. Figure 1b illustrates the location of space and bump on surfaces in a schematic representation. When the bumps are close together and small in size, we expect to observe a small measured acceleration value. An example of such a condition can be seen in the image of the small comb provided in this figure. On the other hand, when the bumps are large and the spaces between them are also large, the built-in accelerometer records a larger value compared with the comb. Examples of swipe surfaces with larger bumps and wider spaces include the breadbasket and the notebook. These selected swipe surfaces should represent a wide range of swipe surfaces with different configurations of the bump-space relationship. The chosen swipe surfaces do not have a binary encoding scheme [4]. The binary encoding scheme uses notches on an acrylic plate to represent the two symbols 0 and 1. The functionality of the notches is the same as a bar code. The sketch in this schema portrays a hand holding a smartphone and wearing

a smartwatch, illustrating the performance on bump-space surfaces. The details of these gestures are discussed in Table 1 in the following section, where we also explore their behaviors.

Table 1. Definitions of different swipe movement behaviors.



Additionally, Figure 1 depicts the correlated swipe gesture paths on each surface. These paths illustrate the sequential arrangement of the bumps and spaces. The starting point is marked by a circle, and the arrow indicates the direction of the swipe gestures. Participants have the freedom to release their index finger at any point along the swipe

path, signaling the end of the swipe motion over the respective surface. The depicted starting point serves as an estimation, as the participants have the autonomy to decide the exact starting point for their swipes.

Next, we explain how to perform swipes on these textured surfaces, focusing on human-determined aspects.

4.2. Human-Determined Aspects

The swipe gesture over textured surfaces consists of three different swipe gesture behaviors: swipe orientation, swipe contact, and swipe movement [2]. Table 1 emphasizes the horizontal and vertical swipe orientations. In the case of a horizontal swipe, the finger aligns with the bump in the same direction, or it slides sideways across the bumps [2]. The top-view illustration in the table clarifies this concept. A horizontal swipe orientation is defined when the finger forms a 90° angle with the bump or when the finger is moved across the surface from a distant position towards the body [2]. The table also outlines the different swipe contacts with the swipe surface, which can be established either with the skin or the nail [2]. The skin-style contact involves swiping over the surface, while the nail-style contact is more akin to scratching on the surface. Both swipe movement styles are shown in Table 1. The swipe movement behavior encompasses swiping with the hand over the surface or solely using the finger. Typically, when participants swipe with their hands, the movement distances are longer compared with when they only use their fingers. When the hand performs the swipe movement, the hand moves while the finger remains rigid, whereas the opposite condition occurs when the finger executes the swipe movement.

Table 1 illustrates the propagation of the vibration signal from the contact with the swipe surface to both mobile devices, namely the smartphone and the smartwatch. Upon contact, the generated vibration is recognized and recorded by both devices. The circle in the table indicates the contact point between the finger and the swipe surface. The snaked line with an arrow represents the transmission of the vibration signal from the index finger to the mobile devices. The vibration signal is transmitted through the skin to both devices. The smartphone is held in the hand while the index finger swipes over the surface, while the smartwatch is worn on the wrist. These two mobile devices serve as replacements for the inertial measurement units (IMUs) used by Han et al. [2] and Shi et al. [56], where IMUs were worn on the wrist and index fingers. In our setup, the smartphone is positioned near the finger, while the smartwatch is worn on the wrist.

Having discussed the human-determined aspects, we now shift our focus to the software-determined aspects in the following section.

4.3. Software-Determined Aspects

Figure 2 presents an overview of the data flow from the built-in acceleration sensor on mobile devices to the classification of the textured swipe surfaces using an SVM. This figure shows the software-defined aspects, including creating a sliding window w_i, applying a filter, using an FFT if necessary to transform the samples from the time domain to the frequency domain, generating the feature vector $\vec{Y}$, and selecting SVM kernels with their parameters.

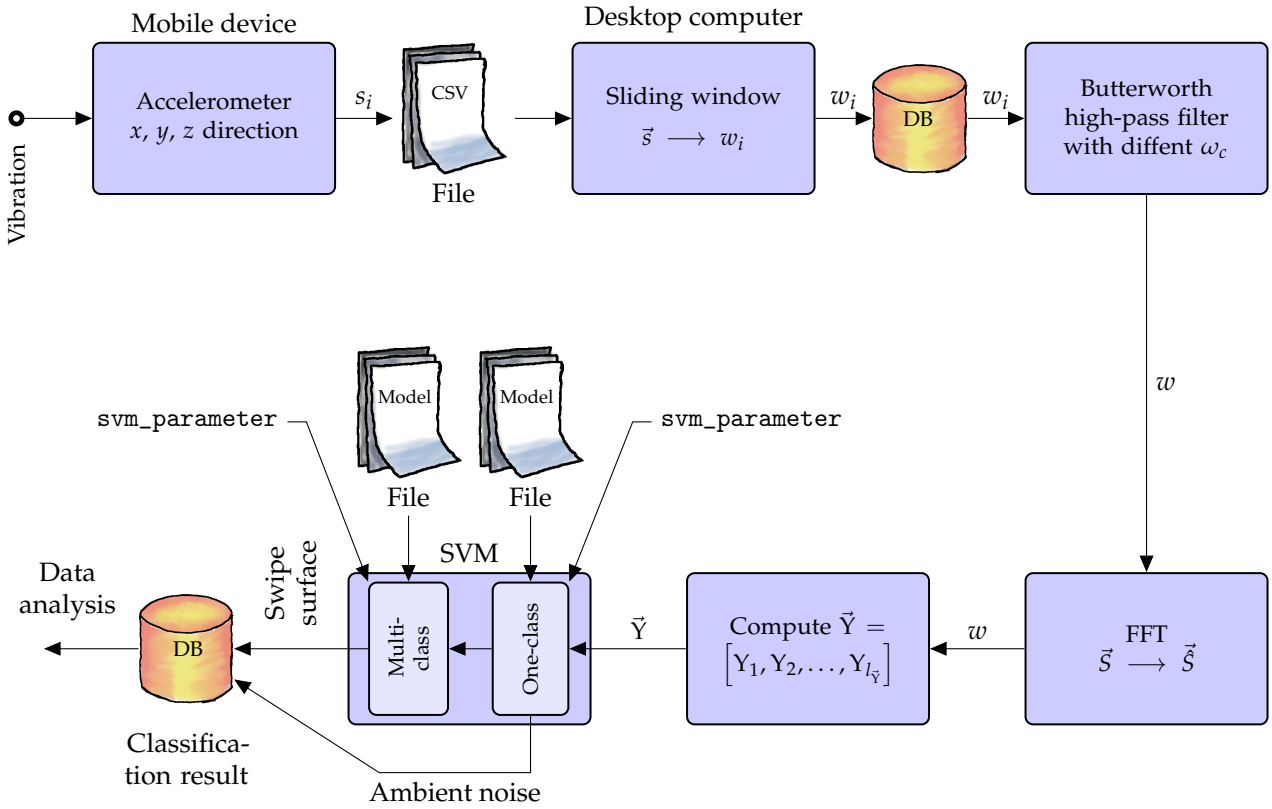


Figure 2. The flow diagram illustrates the process of capturing the vibration signal and classifying it into a feature vector using SVM. It also demonstrates the implementation of the Java library *libSVM* and its corresponding software components. This diagram illustrates the software-determined aspects of the process. In Section 5.1.1, we provide a detailed explanation of the stages shown in the flow diagram. This section focuses on illustrating the process of creating the sliding window w_i, as depicted in Figure 3. The feature elements for $\vec{Y}$ are listed in Table 3, while Table 4 presents the values for ω_c. Additionally, Table 2 displays the `svm_parameter` for the SVM, and the kernels applied for the SVM are outlined in Table 4.

We begin by explaining how the vibration signal is generated when participants swipe over different surfaces. This vibration signal is produced when their fingertip makes contact with the surface and travels from the finger through the hand to the mobile device. The accelerometer in both the mobile device and the smartwatch detects this vibration signal almost simultaneously. When participants press the *Start* button on the smartphone to record the vibration signal, the smartwatch also begins recording as the two devices are connected via Bluetooth. Although the form of the vibration signals may differ, the recording time on both devices is nearly identical.

In our Java Android software application, we utilize the `SENSOR_DELAY_FASTEST` mode to register the accelerometer sensor listener Liu et al. [33]. This selection allows us to efficiently access the accelerometer data.

The structure of the sliding window w was discussed in Equation (1). The timestamp t_i of each sample s_i is recorded in nanoseconds and corresponds to a unique timestamp on the sliding window w. Additionally, we record the swipe surface and swipe behavior alongside each sample and its corresponding measured acceleration in a CSV file.

Once the vibration recording process is completed, we transfer these CSV files from the mobile devices to a desktop computer. The mobile devices are only used to record the vibration signal. The process of training the SVM and predicting the swipe surface is performed on the desktop computer. We chose this setup because of the higher computing power of the desktop computer, which is beneficial for training the SVM under various conditions and reduces computational time.

We developed three Java applications on the desktop computer and one Java Android software application on the mobile devices. The first desktop application handles the sliding windows w_q, the second application is responsible for training the SVM under different conditions, and the last application classifies the different swipe surfaces. The first Java application reads the stored CSV files, splits the recorded samples into sliding windows w, and stores them in a database. The details of this process are explained in more detail in Section 5.1.1.

All the sliding windows w_q are stored in the database, which provides the data for the Java application responsible for SVM training and prediction. The database also stores the number of elements in each feature vector and the corresponding calculations for these elements. The details of the number and computation of these elements are also explained later in Section 5.1.1.

Once the sliding windows w_q are retrieved from the database, the first step is to process the samples within each window. We apply a Butterworth high-pass filter to the sliding windows w_q to prepare them for computing the elements of the feature vector $\vec{Y}$. The Butterworth high-pass filter [57], originally implemented in C++, was converted to Java. We experiment with different cut-off frequencies ω_c for this filter to examine its effect.

The next stage involves applying the FFT to process the samples within each sliding window w_i. However, not every feature vector requires a filtered or frequency-domain representation of w. Hence, the stages involving the Butterworth high-pass filter and FFT can be bypassed depending on the construction of the feature vector. Once the feature vector $\vec{Y}$ is computed, it is used by the SVM from the libSVM library for swipe surface classification.

First, the SVM must be trained to classify different swipe surfaces. After this training process, the SVM is able to classify the swipe surface. The training process of an SVM affects the classification of swipe surfaces. Therefore, in the following discussion, we consider the training process of an SVM and the classification process of swipe surfaces.

Our Java application implements the `svm.svm_predict(model,` $\vec{Y}$`)` method from this libSVM library to classify the swipe surface based on the trained model. Similar to Qin et al. [50], our approach involves using two SVMs in a row. The first SVM, known as the one-class SVM, determines whether $\vec{Y}$ contains samples of swipe surface or ambient noise. The one-class SVM approach [58] is commonly used for anomaly detection, including intrusion detection, medical diagnosis, fraud detection, and surveillance [59]. In the training phase of the one-class SVM, only a feature vector containing samples of swipe surfaces is needed, and it identifies outliers among the samples of the swipe surfaces [60]. It is easier to find appropriate feature vectors that contain only samples of swipe surfaces than typical ambient noise [61]. If the one-class SVM identifies the feature vector as a potential swipe surface, the multiclass SVM classifies the vector further. In addition to our proposed solution, Lukowicz et al. [17] labeled a sliding window as ambient noise, which is unrelated to a swipe surface. DeVaul and Dunn [35] and McGrath and Li [28] trained ambient noise as a separate class in the SVM. They referred to ambient noise as the garbage class [35] or negative samples [28]. In their approach, several hours of ambient data were recorded and used to train the SVM. The drawback of such approaches is the need for proper ambient noise to train the SVM initially. Our proposed approach, depicted in Figure 2, allows us to focus on potential swipe gestures during the training phase. In the prediction phase, the SVM can distinguish between a surface and ambient noise. However, to exclusively train a swipe gesture using the SVM, it is necessary to ensure that only one swipe gesture is trained. This process is discussed in more detail in Section 5.1.1.

After the SVM finishes classifying the swipe surface and ambient noise, the database stores the classification results and the classified swipe surface. Later, we access this data in the database for further analysis using the statistical software program *R*.

Returning to the Java source code of the SVM, the `svm.svm_predict(model,` $\vec{Y}$`)` method from the Java library *libSVM* requires both the `model` variable and the feature vector $\vec{Y}$. Since SVM models are difficult to interpret, we consider them as *black box*

models [62]. At this stage, the swipe surfaces are already trained. So we already have the variable `model`. To obtain the variable *model* for the method `svm.svm_predict(model, Ŷ)`, we have to train the SVM first.

For SVM training, we utilize the `svm.svm_train(Ŷ, svm_parameter)` method from the *libSVM* library. This method provides us with the `model` variable required for the `svm.svm_predict(model, Ŷ)` method. During SVM training, we specify values for the `svm_parameter` variable, which stores the SVM hyperparameters [40,63]. These hyperparameters enable us to configure the various kernels used by the SVM, as summarized in Table 2.

Table 2. SVM hyperparameters used to train the SVM. The SVM hyperparameters are stored in the variable `svm_parameter` and are denoted in typewriter font, enclosed in braces. These hyperparameters are used for the `svm.svm_train(Ŷ, svm_parameter)` method.

Kernel	`cache_size` **in MB**	`degree`	ν (`nu`)	γ (`gamma`)	C (`C`)	ϵ (`eps`)
Linear	300	—	10^{-3}	—	100	0.5
RBF	300	—	10^{-3}	10^{-6}	1250	10^{-3}
Polynomial	300	5	10^{-3}	10^{-6}	100	10^{-3}
Sigmoid	300	—	10^{-3}	10^{-6}	100	10^{-3}

The `svm_parameter` variable contains the hyperparameters `cache_size`, `degree`, ν, γ, C, and ϵ. The corresponding subparameter names for these hyperparameters are listed in the table. The `cache_size` variable determines the amount of memory allocated by the SVM on the desktop computer. We set this variable to 300 MB for all used SVM kernels. Setting the value too low may hinder SVM training, while setting it too high may exceed the available memory. The `degree` column allows us to control the highest exponent of a polynomial kernel function [40,64]. A high degree can result in overfitting, while a low degree may lead to decreased accuracy [64]. The ν variable acts as an upper bound on the fraction of training errors and a lower bound on the fraction of support vectors [10]. The hyperparameter C represents the penalty weight of the error term and must be greater than zero [49,65]. A higher value of C imposes a higher penalty on errors, while a low value lightly penalizes misclassifications and may result in erroneous separation [65]. The ϵ value denotes the tolerance level for the solution to approach zero before the solver stops the iterations [64,66]. Furthermore, the table includes specific values that are only applicable to certain kernels [40]. For instance, the `degree` subparameter is only valid for the polynomial kernel, and γ does not apply to the linear kernel. If a subparameter does not impact the kernel, we indicate it with a line (—) in the table. According to Sangeetha and Kalpana [47], there is no definitive method for determining the appropriate SVM kernel and its hyperparameters. We employed a combination of trial and error along with the suggested values provided by Arbabshirani et al. [40] to determine the hyperparameters. By utilizing the values outlined in Table 2, we were able to successfully classify different solid swipe surfaces.

The samples from the built-in accelerometer were stored on the mobile devices in a CSV file. We download the CSV files from the devices using *Android Studio Chipmunk 2021.2.1 Patch 2* integrated development environment (IDE). To process the downloaded CSV files, we utilized a Java software application developed using the *JSwing* library on the *TravelMate 7740G* laptop. The SVM training was performed using the libSVM library (version 3.24) [10]. The recorded vibration signal data was stored in an SQL database using *SQLite*. During the development of the software application, we observed that training different SVMs required several days. To expedite this process, we employed the *Odroid N2+* single-board computer running the *Linux Mate* operating system. See Appendix B for further technical details.

Method

The laboratory user study proceeded as follows: Participants performed practice swipes under the described swipe conditions, as described in Section 4. Once they felt comfortable, the vibration signals recording began. Participants pressed the *Start* button on the smartphone before swiping over the different surfaces. Both mobile devices recorded the vibration signals, as the smartwatch was connected to the smartphone via Bluetooth. Therefore, the vibration signals were recorded simultaneously on both devices while the participants performed a single swipe. The smartphone displayed the required swipe method, orientation, and surface. Participants pressed the *Start* button again once they finished their swipe movement, after which the data recording process ceased on both devices. The recorded data was stored in a CSV file located on each mobile device. Once a swipe movement was completed, participants proceeded to the next task, with the smartphone displaying the requested swipe surface. The recording process for the next vibration signal began when the participant pressed the *Start* button on the smartphone again. These steps were repeated until all swipe tasks were finished. The user study for each participant concluded with demographic questions, as well as inquiries about their preferred and dispreferred swipe movement behavior and swipe surfaces.

Subsequently, the next participant was included in the user study, covering all possible combinations.

Mobile devices: We used the *Samsung GALAXY Note 3* smartphone and the *Sony SmartWatch 3* smartwatch to capture the vibration signals with a built-in accelerometer. The dimensions of the smartphone are 151.2 × 79.2 × 8.3 mm. The smartwatch has a size of 35.8 × 50.8 × 9.9 mm. Both devices operate on the *Android* operating system. We developed the applications for these mobile devices using *Android Java* within the Android Studio IDE.

Participants: We recruited twelve participants primarily from our university. On average, the participants were 29.00 years old (SD: 4.05), with six female participants. Ten of the participants used their right hand for the swipe movement.

The user study for each participant had an approximate duration of 30 min.

We conducted a four-factor user study with a within-subjects design: 12 (participants) × 5 (swipe surfaces, see Figure 1) × 2 (swipe orientations, see Table 1) × 2 (swipe contacts, see Table 1) × 2 (swipe movements, see Table 1) × 2 (repetitions) × 2 (mobile devices) = 1920 swipe movements.

Using the recorded vibration data from this user study, we evaluate the software-determined aspects. The specific aspects considered are discussed in the next section.

5. SVM Training and Classification

Following the user study, accelerometer samples were collected from each mobile device, primarily addressing human-determined aspects. Our focus now shifts to software-determined aspects, specifically involving SVM training and classification of vibration signals generated by participants during the user studies.

5.1. Screening Phase

In the screening phase, we aim to distinguish between factors that are relevant to human-determined and software-determined aspects, and those that are not.

5.1.1. Training

We can train the SVM with the `svm.svm_train(`$\vec{Y}$`, svm_parameter)` method from the libSVM. Figure 3 shows how we obtained the sliding windows and how we built $\vec{Y}$.

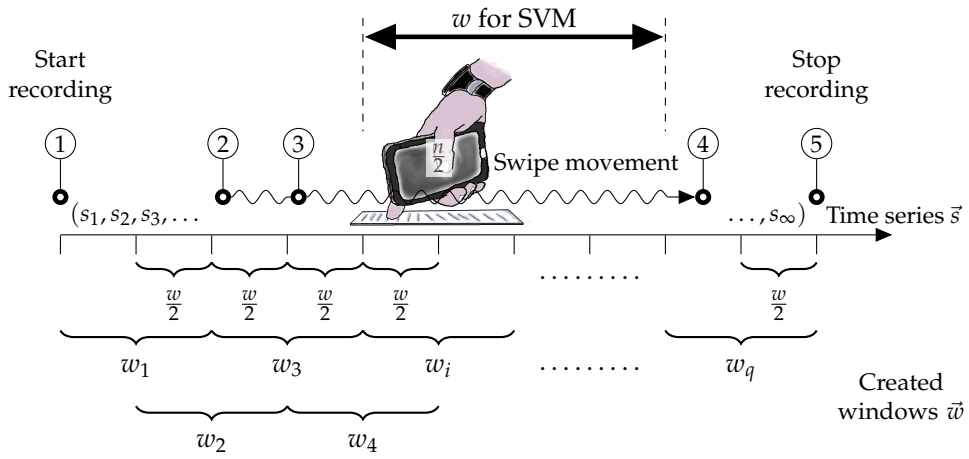


Figure 3. Stages one to five illustrate ①–⑤ the process of performing a swipe gesture over a textured surface. From this time series $\vec{s}$ sliding windows ws are extracted. Two ws from each swipe movement are used to train the SVM on the swipe surface. One w is used to classify the swipe surface.

We created sliding windows, denoted as w_q, from a time series $\vec{s}$. The length l of w_i is 32, 64, or 128 samples, which is suitable for the FFT. The FFT results are then applied to generate the feature vector $\vec{Y}$. Han et al. [2] used a sliding window length of 64 samples, so we selected additional 32 and 64 samples to explore the effects of smaller and larger sliding windows. We created sliding windows by segmenting the remaining time series $\vec{s}$. We constructed the sliding windows using the overlap-add method. Thus, $w_i = \left[w_{(i-1)}/2, w_i/2\right]$, where $w_{(i-1)}/2$ contains samples from the previous half window, and $w_i/2$ is filled with the next vibration signals to complete one sliding window. Following this approach, we obtained q sliding windows denoted as $\vec{w} = \{w_1, w_2, \ldots, w_i, \ldots, w_q\}$.

Now, we want to explain considered samples of a swipe movement. Liu et al. [33] advised that participants keep their hands static at the beginning and end of the recording to ensure optimal performance of the swipe gesture on these surfaces. This practice helps to avoid the presence of unobservable acceleration signals, which could have a negative impact on the SVM training process [65,67]. Pan et al. [16] explain that a swipe movement can be divided into three parts: an impulse-like signal, a stick-and-slip part, and the removal of fingers from the surface. This stick-and-slip phase represents the general form of friction-induced vibrations. In Figure 3, the stick-and-slip part is illustrated in ③ and ④. Liu et al. [33] captured the stick-and-slip part while they recorded the vibration signal of the swipe gestures. We want to exclude this effect in the data set to train the SVM. Therefore, we focused on sliding windows that were close to the center of all generated windows during a swipe movement on a given surface.

It is important to consider the presence of label and feature noise [8]. Label noise occurs when a sliding window is assigned an incorrect label for the corresponding swipe surface. On the other hand, feature noise occurs when participants perform swipe movements that deviate from the requested behavior during the user study. Such feature noise occurs also when we consider the impulse-like signal for the sliding window. Pelletier et al. [8] outlined the impact of low levels of random label noise, up to 25% to 30%, on the classification performance of different swipe surfaces is minimal. However, higher levels of label noise can significantly reduce the classification performance.

Figure 4 illustrates the stages involved in the vibration signal recording process of the smartphone and smartwatch separately. In this figure, the three acceleration directions x, y, and z are depicted. These plots are aligned with the representation of the sliding window w in a matrix structure, as described by Equation (1).

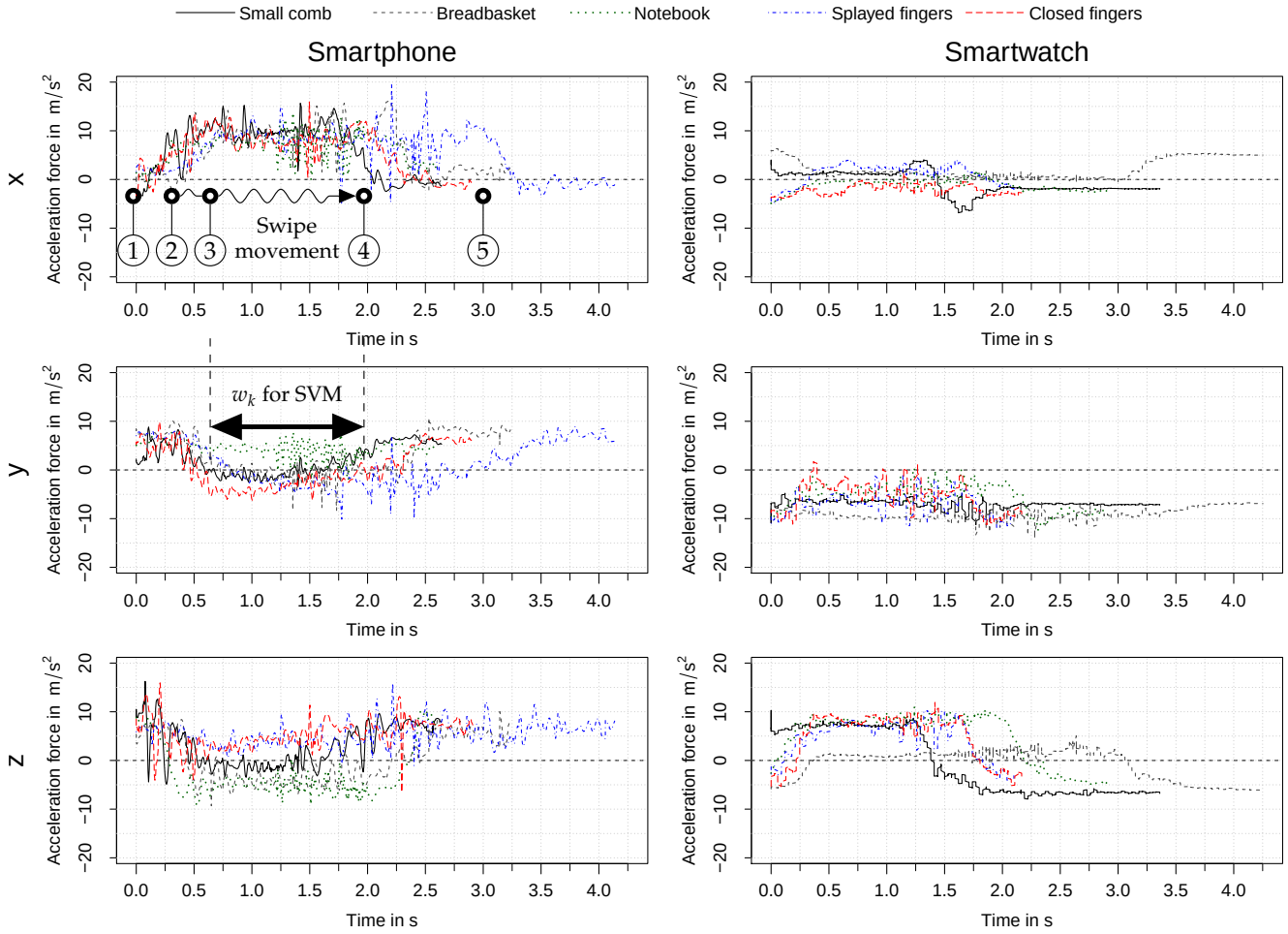


Figure 4. Vibration signals when a participant swipes over the swipe surface while the hand is moving. The first plot highlights again the five stages of a swipe gesture. The applied swipe contact is the nail, and the swipe orientation is horizontal. These last two conditions are fixed during the recorded time samples.

In Figure 4, the swipe contact used is consistently nail, and the swipe orientation is specified as horizontal. Additionally, we can observe from this figure that each swipe surface exhibits a different vibration signal amplitude. This figure highlights the varying swipe behaviors that result in different vibration signal amplitudes. These distinct amplitudes should contribute to different elements in the feature vector $\vec{Y}$, thereby facilitating the accurate classification of swipe surfaces.

Let us now discuss the detailed stages of a swipe movement. The stick-and-slip part occurs between steps ③ and ④ in Figure 4, and this period corresponds to the sliding window w_i used for training the SVM. During stages ① to ③, the finger moves towards the designated swipe object, which typically takes around one second. Between ④ and ⑤, the index finger releases the swipe surface, and at ⑤, the vibration signal recording stops. We can extract the middle part of each recorded swipe movement since the vibration signal exhibits approximate symmetry. In this middle part, we observe a repetitive pattern in the vibration signal. We can imagine an imaginary straight line that closely aligns with this vibration signal. For instance, when examining the y and z acceleration axes in the plots of the smartphone with the swipe surface *small comb*, the vibration signal hovers around the y-axis value of 0. We discuss further details of the vibration signal in Appendix C.

We encountered at least 12 sliding windows per swipe movement. Therefore, we consider three sliding windows created from the sliding window series $\vec{w}$, which are considered samples nearly in the center of each swipe movement. We trained the SVM with 2/3 of all created sliding windows $\vec{w}$, while the remaining 1/3 of $\vec{w}$ is used to predict the swipe surfaces. This approach follows the train/test split method [60]. Two w_i per swipe movement raised concerns that we might not have enough sliding windows to train the SVM. However, a train/test split with a low number of $\vec{Y}$ gives results comparable to 1000 sliding windows [63]. We also used the train/test split for simplicity. The selection of 2/3 of the sliding windows $\vec{w}$ for training the SVM was done randomly from the set of all three sliding windows w created. The train/test split method was also used to find proper hyperparameters, which was displayed in Table 2. Vabalas et al. [63] explain K-fold cross-validation and nested cross-validation as additional approaches for validating the model of the SVM, which were not applied in this work.

We used the train/test split method to find the suitable hyperparameters, shown in Table 2. This method was used on the data, for instance, where participants swiped horizontally on the surface, as seen in Table 1. After finding the hyperparameters for the SVM in a few conditions where we saw a high classification accuracy, we used these parameters for all conditions.

We constructed several feature vectors $\vec{Y}$ based on a single w_i, represented in Table 3.

Table 3. Training the SVM with different feature vectors $\vec{Y}$. The index ξ denotes the different built-up feature vectors.

ξ	Structure	Ψ	$\{\Xi_1, \ldots, \Xi_n\}$
(1)	$\{\hat{\Xi}, \hat{S}\}$	18	mean, SD, skewness, kurtosis, index distance between $\min(w_i)$ and $\max(w_i)$, and the index distance between two $\max(w_i)$ values
(2)	$\{\Xi, S\}$	18	mean, SD, skewness, kurtosis, index distance between $\min(w)$ and $\max(w)$, and the index distance between two $\max(w)$ values
(3)	$\{\Xi, \hat{\Xi}\}$	10	$\Xi \rightarrow \sum_{i=1}^{3} \text{PSD}_i$, $\|w\|_1$, $\|w\|_\infty$, $\|w\|_F$, $\min(w)$, $\max(w)$ $\hat{\Xi} \rightarrow \sum_{i=1}^{3} \text{PSD}_i$, $\|w\|_1$, $\|w\|_\infty$, $\|w\|_F$
(4)	$\{\Xi\}$	4	$\sum_{i=1}^{3} \text{PSD}_i$, $\|w\|_1$, $\|w\|_\infty$, $\|w\|_F$
(5)	$\{\Xi\}$	6	$\sum_{i=1}^{m=3} \text{PSD}_i$, $\|w\|_1$, $\|w\|_\infty$, $\|w\|_F$, $\min(w)$, $\max(w)$
(6)	$\{\hat{\Xi}\}$	$3 \cdot l$	mels_x, mels_y, and mels_z
(7)	$\{\hat{S}\}$	$3 \cdot l$	$\hat{S}_x, \hat{S}_y, \hat{S}_z$

The *structure* column in Table 3 illustrates the elements considered in $\vec{Y}$ and indicates the domain in which each element is created, denoted by $\in \{\text{time}, \text{frequency}, \text{both}\}$. The notation follows Equations (2)–(4). For example, $\hat{S}_{3,l}$ can be represented as $\left\{\hat{S}_{1,1}^{(x)}, \ldots, \hat{S}_{3,l}^{(z)}\right\}$, where the index $i \in \{1, 2, 3\}$ corresponds to the three acceleration directions x, y, and z, and l indicates the number of samples in a sliding window w_i, which can be 32, 64, or 128 samples. The Ψ column in the table represents the sum of evaluated characteristics in $\vec{Y}$. Ψ does not represent $l_{\vec{Y}}$.

Feature vectors with indices three to six consist only of computed characteristics. The table's column $\{\Xi_1, \ldots, \Xi_n\}$ lists the applied characteristics that describe the vibration signal of a surface. The other remaining indices include samples from a single w_i. Indices one and seven have $\hat{S}$ in the frequency domain, while index two has S in the time domain.

To describe the vibration signal of a textured surface, Miluzzo et al. [32] proposed the 1-norm, infinity norm, and Frobenius norm of w. These norms capture the vibration signal behavior for all three acceleration directions within a single element of $\vec{Y}$. For other characteristic calculation approaches, the computation is performed separately for each direction. The 1-norm is denoted as $\|w\|_1$, the infinity norm as $\|w\|_\infty$, and the Frobenius norm as $\|w\|_F$. We utilize a $3 \times l$ matrix w from Equation (1) to calculate these norms using Equations (5)–(7).

$$\|w\|_1 = \max_{1 \le j \le l} \sum_{i=1}^{l} |S_{i,j}| \tag{5}$$

$$\|w\|_\infty = \max_{1 \le i \le 3} \sum_{j=1}^{l} |S_{i,j}| \tag{6}$$

$$\|w\|_F = \sqrt{\sum_{i=1}^{3} \sum_{j=1}^{l} |S_{i,j}|^2} \tag{7}$$

The norm can be computed in both the time domain and the frequency domain. In the time domain, a single element in w_i is denoted as $S_{i,j}$. If we replace $S_{i,j}$ with $\hat{S}_{i,j}$, the norm is computed in the frequency domain. This same definition also applies to the calculation of skewness, kurtosis, and PSD, as shown in Equations (8)–(10). Equation (10) [68] is also named as *mean square amplitude*.

$$\text{Skewness}_{i=(xyz)} = \frac{\frac{1}{l} \sum_{j=1}^{l} \left(S_{i,j} - \bar{S}_{i,j}\right)^3}{\left[\frac{1}{l-1} \sum_{j=1}^{l} \left(S_{i,j} - \bar{S}_{i,j}\right)^2\right]^{3/2}} \tag{8}$$

$$\text{Kurtosis}_{i=(xyz)} = \frac{\frac{1}{l} \sum_{j=1}^{l} \left(S_{i,j} - \bar{S}_{i,j}\right)^4}{\left[\frac{1}{l} \sum_{j=1}^{l} \left(S_{i,j} - \bar{S}_{i,j}\right)^2\right]^2} - 3 \tag{9}$$

$$\text{PSD}_{i=(xyz)} = \frac{1}{l} \sum_{j=1}^{l} |S_{i,j}|^2 \tag{10}$$

$$\text{Mels}_{i=(xyz)} = 2595 \cdot \log_{10}\left(1 + \frac{\hat{S}_{i,j}}{700}\right) \tag{11}$$

The index i in Equations (8)–(11) represents the three acceleration directions: x, y, and z. When computing the previously defined 1-norm using Equation (5), the feature vector $\vec{Y}$ consists of only one element, resulting in a length l of one for $\vec{Y}$. However, the situation is different when applying Equations (8)–(11). For example, when computing the skewness using Equation (8), the feature vector has a length of three. This is because the calculation is performed for each acceleration direction (x, y, and z), yielding separate results for each direction. Equation (11) describes the calculation of mels, which are part of the mel scale used in MFCC computation. We specifically selected mels with low computational effort for the feature vector. We did not require frequency information over time. To compute mels, we first transformed the samples within w_i to the frequency domain, denoted by the symbol $\hat{S}$ in Equation (11). After this transformation, Equation (11) was applied. As mentioned before, Ψ represents the number of characteristics within one $\vec{Y}$. Since mels are calculated for the entire sliding window w_i, Ψ equals the size of w_i. Therefore, the value of l in the Ψ column of the table is 32, 64, or 128 samples.

The length $l_{\vec{Y}}$ of $\vec{Y}$ depends on the computed characteristics Ξ and the added samples $\hat{S}_{1,1}, \ldots, \hat{S}_{3,l}$ from w_i. For example, if we consider $\vec{Y}_{\xi=1}$ and the number of samples within a sliding window is 32 samples, the feature vector has a length of 114 elements It is because $\hat{\Xi}$ has 18 elements, and $\hat{S}_{1,1}, \ldots, \hat{S}_{3,l}$ is $3 \cdot l = 3$ acceleration directions $\cdot$ 32 samples $= 96$ elements $= l_{\vec{Y}}$. Therefore, the length of the feature vector $l_{\vec{Y}}$ can be compared with the reported length of $\vec{Y}$ by Han et al. [2], who used 109 elements for the feature vector. The lengths are nearly the same in both cases.

In Table 3, we used $3 \cdot l$ to indicate that we used the three recorded acceleration directions from the sensor. Based on the aforementioned considerations, the selected feature vectors have lengths ranging from 4 to 402 elements in Table 3.

Table 4 summarizes the necessary independent factors and their associated labels for the complete experimental design. The response or dependent variable was the correct classification of different swipe surfaces.

We used the same kernel, RBF, for both the one-class SVM and the multiclass SVM. Inspiration from Le et al. [9] and Qin et al. [50] led us to apply the RBF kernel for separating the vibration signal from the ambient noise. Since participants had difficulty performing the different requested swipe contacts and swipe movements and the swipe movement behavior showed minimal differences, we treated the swipe contacts and swipe movements as one unit. Although Harrison et al. [4] and Han et al. [2] discussed swipe contacts with the term *nail*, they did not address the input issue with fingernails. Other reasons for this were that the smartphone, as shown in Table 1 could not be held particularly well with small hands. Based on this consideration, we labeled these two swipe movement behaviors as both in the table and make no distinction between these two swipe contacts as in Table 1 described. If we label another factor's level as both, it means we included all recorded data from that factor.

Table 4. Applied factors and levels in the screening phase.

	Factor	Levels
Software	l in samples:	32, 64, 128
	ω_c in Hz:	0, 0.1, 5, 20
	$\vec{Y}$:	seven different feature vectors; see Table 3
	One-class kernel:	same as multiclass kernel or RBF
	Multiclass kernel:	linear, RBF, polynomial, sigmoid
	Mobile device:	smartphone, smartwatch, both
Hum.	Swipe contact:	both
	Swipe orientation:	horizontal, vertical, both; see Table 1
	Swipe movement:	both

For these experiments, we consider the following factors: 3 (samples) $\times$ 4 (ω_c) $\times$ 7 ($\vec{Y}$) $\times$ 2 (one-class kernels) $\times$ 4 (multiclass kernels) $\times$ 3 (mobile devices) $\times$ 1 (swipe contacts) $\times$ 2 (swipe orientations) $\times$ 1 (swipe movement) $=$ 4032 experiments.

However, conducting all 4032 experiments can be impractical due to their large number and the associated time and cost [69]. To address this issue, we employed a D-optimal design, which allows us to reduce the number of experiments while still obtaining reliable results in a shorter time frame. Optimal designs offer a solution by allowing us to maintain comparable or even superior statistical power in detecting the effects of interest while reducing the number of experimental runs required [69,70]. Despite their benefits, optimal designs are currently underutilized in screening experiments and have received limited attention in the field of software testing [71]. However, they have been successfully applied in diverse research domains, including pharmacy [72,73], mechanical engineering [74,75], and psychology [70].

After obtaining the 300 experiments from the D-optimal design, see the calculation details of D-optimal design in Appendix D, we proceeded to train the SVM, which took approximately one day. Following the training phase, we utilized the trained SVM to classify the various swipe surfaces, a process that was completed within a few hours. With the experiments now concluded, we proceed to the classification of the swipe surfaces.

5.1.2. Classification

With the `svm.svm_predict(model,` $\vec{Y}$`)` method from the Java library *libSVM* we are able to classify the swipe surfaces explained in Figure 1. Because we ran 300 experiments, we obtained 300 models in the training phase of the SVM.

If the one-class SVM classifies $\vec{Y}$ as ambient noise, it is not taken into account when determining the classification accuracy. Currently, we assume that ambient noise is always classified correctly. In Section 6.3, we evaluate the one-class SVM.

Our objective is to identify conditions where the SVM detects nearly 100% of the swipe surfaces. We focus on software aspects that suggest a potential for high classification accuracy across all swipe surfaces. We exclusively utilize 64 and 128 samples within each sliding window, as the 32 samples in w_i do not yield a high classification accuracy. Also, we consider only the swipe orientations horizontal and vertical. We train the SVM using all recorded data for the swipe contact and swipe movement. With these considerations, we proceed to the detailed phase.

5.2. Detail Phase

In this phase, our primary focus is on factors that offer a classification accuracy of 90% or higher for the swipe surfaces, as discussed in the previous section.

5.2.1. Training

In the detailed phase, we used the acceleration sensor data recorded during the laboratory user study to train the SVM, keeping the human-determined aspects. However, the software-determined aspects were adjusted, focusing on conditions expected to yield high classification accuracy for textured surfaces. The software application and procedures remained consistent with those employed in the screening phase.

5.2.2. Classification

Table 5 presents the selected candidates along with the corresponding conditions that resulted in the highest and lowest classification accuracy for the different swipe surfaces. If the one-class SVM classifies a feature vector as ambient noise, it is not included in the computation of correct classification in the table.

Table 5. The selected conditions for the SVM for the evaluation. The short form *cont.* indicates the swipe contact on the different swipe surfaces. The abbreviation *orient.* represents the swipe orientation. $\overline{\text{Acc.}}$ indicates the correct classification of the different swipe surfaces. In the software-determined section in this table, $\vec{Y}$ link to the index ξ in Table 3. We use these six different conditions for further evaluation in this work.

	Factor	Both Devices		Smartphone		Smartwatch	
		Best	**Worst**	**Best**	**Worst**	**Best**	**Worst**
Software	l in samples:	128	128	128	64	128	128
	1-class:	RBF	Sigmoid	RBF	RBF	RBF	RBF
	m-class:	RBF	Sigmoid	RBF	RBF	RBF	RBF
	ω_c in Hz:	20	0.1	0.1	20	20	0.1
	$\vec{Y}$:	(6)	(6)	(6)	(3)	(7)	(4)
	$l_{\vec{Y}}$:	384	384	384	10	384	4
Hum.	Contact:	both	both	both	both	both	both
	Orient.:	horizontal	vertical	vertical	horizontal	horizontal	horizontal
	Move.:	both	both	both	both	both	both
	Correct w_i:	104	0	94	58	95	0
	Wrong w_i:	69	190	0	1	0	35
	Noise w_i:	17	0	2	37	0	61
	$\overline{\text{Acc.}}$ in %:	69.61	15.78	97.59	42.03	99.79	20.00

This table shows that the best condition achieves nearly 100% classification accuracy for each mobile device separately. When using data from both the smartphone and smartwatch together to train the SVM, the classification accuracy is still 69.61%, as indicated in the *both devices* column of the table. The worst condition exhibits a lower classification accuracy, but it is never below 15%. The accuracy of the best selected conditions is in the same range as reported by Harrison et al. [4] and Han et al. [2]. To the best of our knowledge, we did not find the worst-selected conditions reported in the literature.

In the following section, we evaluate how the classification accuracy changes when using data from the entire swipe movements or when participants perform swipe movements slightly differently from those in the laboratory user study.

6. Evaluation

We obtained six different SVM models, detailed in Table 5. These classification accuracies were obtained under laboratory conditions. During the laboratory user studies, participants were instructed on how to swipe over the different surfaces. However, in real-world scenarios, users may swipe differently without any guidance, which could potentially affect the classification accuracy. To explore these variations and identify potential usability issues outside of controlled laboratory conditions, we conducted additional user studies in the field. In the following, we discuss the differences between these two types of user studies and the motivation behind conducting them.

Kjeldskov and Stage [76] state that field-based evaluations are often considered essential for assessing the usability of a mobile system. Baillie and Schatz [77] further emphasized that conducting user studies solely in the laboratory or field is insufficient, as both settings are necessary to uncover crucial usability issues. While some studies have questioned the validity of laboratory evaluations compared with field evaluations [78], it is widely accepted that laboratory environments and field studies complement each other [78]. Laboratory studies are valuable for evaluating the application, while field studies are necessary to validate the results [78].

To ensure usability issues are not overlooked, de Sá and Carriço [79] recommended conducting user studies in realistic settings with specific details, as relying solely on laboratory or field tests may lead to missing important insights. Duh et al. [80] highlighted that laboratory and field user studies often uncover different issues and usability problems. While laboratory studies offer advantages such as reduced difficulties in data collection, they cannot address the factors and issues that arise in real-world field settings [80].

However, evaluating usability in the field is not without challenges, as pointed out by Kjeldskov and Stage [76]. Establishing realistic studies that capture key use-context situations, applying established evaluation techniques like observation and think-aloud, and collecting data can be complex in field evaluations due to the multitude of unknown variables and limited control over the environment [76]. In contrast, these difficulties are significantly reduced in laboratory settings [76].

After discussing these two user study types, let us clarify the *whole swipe* condition for evaluating the SVM, representing a real-world scenario.

6.1. Whole Swipe

We used the middle part of a recorded swipe movement to train the SVM using the train/test split approach that was described in Section 5.1.1. This means that we applied the vibration signal data set from the laboratory user study. In the current scenario, the SVM is expected to classify the entire recorded swipe movement, which includes the stages ① through ⑤ shown in Figure 4. Now, w_i could contain a potential swipe surface or ambient noise. The one-class SVM should filter out those w_i that do not correspond to swipe surfaces.

From the complete swipe over a surface, we collected data for $l = 64$ samples, resulting in $n(\vec{w}) = \{9079\}$, and for $l = 128$ samples, the values were $n(\vec{w}) = \{4252, 6725.80, 9519\}$. The symbol $n(\vec{w})$ represents the number of sliding windows $\vec{w}$ extracted from the recorded vibration signals. This symbol is also known as the *cardinality* in mathematics, indicating the number of w in the set $\vec{w}$. The first value in the braces denotes the minimum count of recorded $\vec{w}$, the second value represents the mean count of $\vec{w}$, and the last value indicates the maximum count of sliding windows extracted from all recorded swipe movements. We summarize the aforementioned description of $n(\vec{w})$ in the representation $\{\min(n(\vec{w})), \bar{n}(\vec{w}), \max(n(\vec{w}))\}$, where $\min(n(\vec{w}))$, $\max(n(\vec{w})) \in \mathbb{N}$ and the average sliding window count $\bar{n}(\vec{w}) \in \mathbb{R}$.

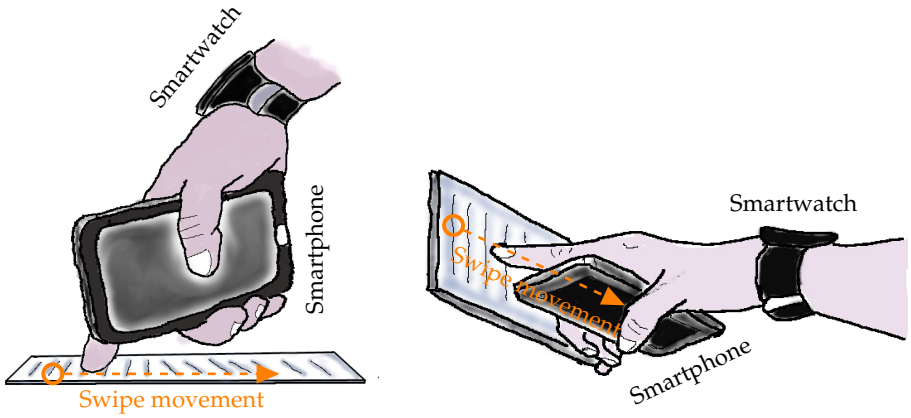
A sample size of $l = 64$ was utilized to record vibration signals from the smartphone, giving rise to a single value for this particular condition. Consequently, the minimum and maximum counts of $\vec{w}$ were eliminated. The value of l for this condition is presented in Table 5. For conditions such as *both devices* in the best and worst scenarios, or *smartwatch* in the best and worst conditions, a sliding window w of $l = 128$ samples was utilized. These l values are also indicated in Table 5. Since there are five conditions with $l = 128$ samples in this table, we were able to determine $\min(n(\vec{w}))$, $\bar{n}(\vec{w})$, and $\max(n(\vec{w}))$. However, it is not feasible to use this notation for $l = 64$ samples since we have only one condition for this scenario.

In the condition *both devices* in Table 5, where vibration signals from both the smartphone and the smartwatch were combined, the maximum count value for $l = 128$ is typically higher than for $l = 64$. When sliding windows are obtained from both devices, approximately double the minimum value.

With this definition provided, we now move on to the in-field user study.

6.2. In-Field User Study

Under the typical usage of vibration as an input technique, the human- and software-determined aspects are not known in advance. While we have control over the software aspects, the human-determined aspects are influenced by individual preferences and behaviors. To account for this variability, we allowed the participants to freely swipe over the different surfaces without providing specific instructions on how to perform the swipes. They had the flexibility to choose the swipe orientation, swipe contact, and swipe movement, as shown in Figure 1. We provided the participants with the mobile device and the swipe surface, and they performed the swipes in both a sitting and standing position. The standing position was chosen to simulate scenarios such as using a light switch. The concept is depicted in Figure 5.



(a) Swipe movement during sitting (b) Swipe movement during standing

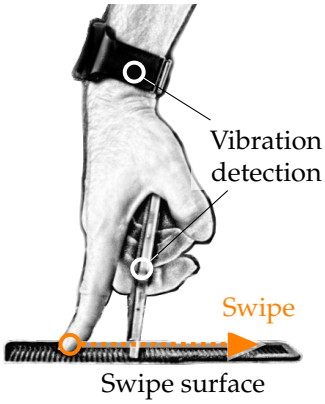
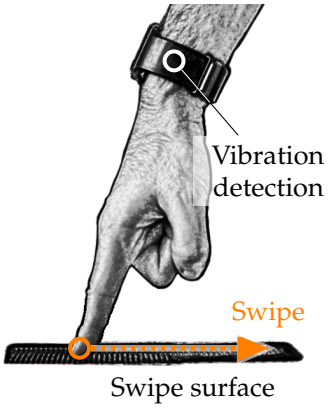
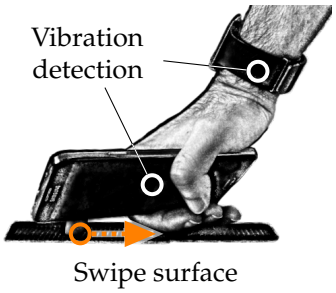
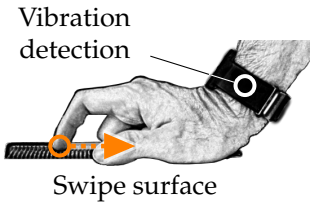
Figure 5. Swiping with different movement behaviors on the surface.

If the placement of the swipe surface is changed, as illustrated in Figure 5b, the SVM will encounter untrained feature vectors. In this study, we aimed to investigate the impact of swipe surface classification accuracy under such conditions. Figure 5a shows the same swipe surface placement as in the laboratory user study.

Next, we explain the swipe movements represented in these figures. Figure 5a illustrates the vertical swipe orientation, while Figure 5b depicts the horizontal swipe orientation. Both images demonstrate the hand movements corresponding to Table 1.

During the recording of vibration signals in laboratory user studies, we used both mobile devices. However, when participants used only the smartwatch, their hand posture changed. Table 6 illustrates this alteration in hand posture.

Table 6. Variations in hand pose when employing both mobile devices or relying solely on the smartwatch for detecting vibration signals.

	Both Mobile Devices	**Smartwatch**
Moving hand		
Moving finger		

Under this condition, the swipe contact on the swipe surfaces is always made with the finger, regardless of the swipe direction. In all images shown in the table, the swipe direction is always vertical. The difference is in the number of vibration detections. The left column of the table shows the hold of the two mobile devices used in the laboratory user study. The right column shows the condition when only the smartwatch is used to detect the different swipe surfaces. The difference between these two conditions is the change in hand posture. In the right column of the table, the hand posture looks like writing with the finger [81] or swipe over different swipe surfaces [2,4].

Method

For the in-field user study, each participant could choose between the smartphones *Samsung GALAXY Note 3* and *LG P700* and was assigned the smartwatch *Sony SmartWatch 3*. We want to figure out which smartphone size the participants prefer in this user study. To compare preferences, we have selected the *Samsung GALAXY Note 3* and the *LG P700* smartphone. The Samsung GALAXY Note 3 is larger than the LG P700. As we are only interested in smartphone size, we have provided all participants with the same smartwatch for the study.

Before recording the vibration signals, the participants were asked to demonstrate how they would swipe over the surfaces. They had the opportunity to practice and choose a swipe behavior they felt comfortable with. The examiner then stored the selected swipe movement behavior on the mobile device, including swipe orientation, swipe contact, and swipe movement. Once the selection was made, participants were required to stick with their chosen swipe movement behavior throughout the study. In one part of the study, half of the participants started swiping in the sitting condition while the other half started in the standing condition, or vice versa. Figure 5 illustrates the differences between these swipe conditions. The participants initially used both mobile devices, and then they proceeded to

use only the smartphone or the smartwatch to record the vibration signals. We provided different combinations to ensure that participants did not always start with both devices. The order of the swipe surfaces was randomly assigned to the participants, and the order of movement behaviors changed after each participant.

During the screening phase of the study, several differences were identified compared with the already conducted user study:

1. Selection of swipe behavior: In the screening phase, the examiner provided instructions to participants regarding the swipe behavior to be used in the study.
2. Different swipe conditions: During the screening phase, participants only performed swipes in the sitting condition. However, in the in-field user study, both sitting and standing swipe behaviors were included. The change in swipe conditions is significant because the position of the mobile devices and the swipe surface differs when standing, as depicted in Figure 5. This variation in the swipe condition should answer the question of whether and how it affects the classification of the swipe surface.
3. Bluetooth connection: In the screening phase, both mobile devices were consistently connected via Bluetooth. However, in the in-field user study, the smartphone and smartwatch were used separately, without a Bluetooth connection between them. Participants performed swipe movements using both connected and unconnected devices. When there is no Bluetooth connection, the hand posture changes, as illustrated in Figure 6.

Mobile devices: We used the smartphone and smartwatch from the laboratory user study. We added an *LG P700* smartphone to this user study. The dimensions of the device are 125.5 × 67 × 8.7 mm. All three mobile devices run the *Android* operating system, so we applied the coded software directly from the laboratory user study.

Participants: We invited twelve participants from our university. The average age of the participants was 28.33 (SD: 3.70) years, and six of them were female. During the study, eleven of the participants swiped over the swipe surface with their right index finger. The user study took approximately 20 minutes for each participant.

We conducted a three-factor user study with a within-subjects design, considering only the factors we can control. These controlled factors are movement behaviors $\in \{\text{sitting}, \text{standing}\}$, the mobile devices $\in \{\text{smartwatch}, \text{smartphone}, \text{both devices}\}$, and the five different swipe surfaces. These factors lead to 12 (participants) × 2 (movement behaviors, see Figure 5) × 3 (mobile devices) × 5 (swipe surfaces, see Figure 1) × 2 (repetitions) = 720 swipe movements. We used all the extracted sliding windows $\vec{w}$ from these swipe movements.

We observed in the first two participants that the repetition of the task was not displayed to them. After checking the data, we realized that they swiped over the swipe surfaces for a longer duration, resulting in more extracted $\vec{w}$ than expected. Additionally, two other participants started and stopped the vibration recording for one swipe movement behavior extremely quickly, resulting in no recorded vibration signals. These examples highlight the high variance in swipe lengths. Nevertheless, despite these challenges, we were still able to collect sufficient data in the user study, making data analysis possible. For $l = 64 \rightarrow n(\vec{w}) = \{715, 760.50, 834\}$ and for $l = 128 \rightarrow n(\vec{w}) = \{275, 1075.05, 1441\}$, we only distinguished between the mobile device types smartwatch and smartphone, without considering different manufacturers.

6.3. Ambient Noise User Study

Previously, the one-class SVM was tested only on sliding windows that contained potential vibration signals. To evaluate the performance of the one-class SVM, we focused on human physical activities that other researchers [35,36,82–85] aimed to detect using accelerometers. These activities included walking, climbing up stairs, climbing downstairs, and typing on a desktop computer keyboard [36,82–85]. Based on their considerations, we extracted several usage conditions for the smartphone and the smartwatch that were

unrelated to potential swipe surface vibration signals. These vibration signals were named ambient noise.

Method

The activities and the number of sliding windows w extracted from each recorded ambient noise task are listed below. During the user study, both mobile devices recorded ambient noise while the participants performed these activities. The activities performed by the participants are listed below.

1. Walking: The participants walked around our university for about five minutes. They wore the smartwatch on their wrist and held the smartphone in their hand. This walking activity also involved climbing up and down stairs. We included this condition based on our observation of how people typically hold their smartphones while walking on the campus at our university.
   $l = 64; n(\vec{w}) = \{1903\}$;
   $l = 128; n(\vec{w}) = \{951, 1525, 2144\}$;
2. Text typing: The participants picked up the smartphone from the desk and typed a text. Afterward, they placed the smartphone back on the desk. We did not provide a specific input text, and it was not crucial how they typed the text, that is, whether it was with their index finger or thumb, for example. The participants repeated this task three times.
   $l = 64; n(\vec{w}) = \{892\}$;
   $l = 128; n(\vec{w}) = \{445, 811.80, 1126\}$;
3. Phone call: We simulated a phone call for the participants. In this scenario, they picked up the smartphone from the table. Next, they swiped over the touchscreen to accept the phone call. After a while, they returned the smartphone to the table. The participants repeated this task three times.
   $l = 64; n(\vec{w}) = \{169\}$;
   $l = 128; n(\vec{w}) = \{84, 176, 241\}$;
4. Whole day: The mobile devices recorded vibration signals for a duration of two hours. During this time, the devices were also charging. Each task was repeated twice, allowing us to collect vibration signals for approximately eight hours. These tasks were distributed throughout the day to simulate typical mobile device usage patterns. These charging times are in consideration of the limited battery capacity of the smartwatch. We handed over the mobile devices to the participants and were not able to observe their activities during the recording time.
   The start of the vibration recording varied slightly because the mobile devices were not connected via Bluetooth. Participants pressed the start button on each device to begin recording the vibration signals. A time series of $\vec{s} = \{s_1, s_2, s_3, \ldots, s_{l=128}\}$ was recorded randomly. To manage the data effectively, the software application on the mobile devices stopped after two hours, resulting in nearly 1000 of $\vec{s}$. At this stage, $\vec{s}$ is not converted to a sliding window. When about 500 such time series in one hour have been recorded in one hour, the software application will stop recording the vibration signals until the next hour is started. After the recording time, the sliding windows w with 64 or 128 samples were applied as specified in Table 5 and explained in Figure 3.
   $l = 64; n(\vec{w}) = \{9079\}$;
   $l = 128; n(\vec{w}) = \{4252, 6725, 9519\}$;

Mobile devices: The *LG P700* smartphone was used for the conditions walking, text typing, and phone call, as participants preferred its smaller size, making it easier to hold compared with the Samsung GALAXY Note 3. For the *whole day* condition, the smartphone *Samsung GALAXY Note 3* was preferred due to its longer battery life compared with the LG P700. The smartwatch *Sony SmartWatch 3* was chosen for all conditions in this user study.
Participants: For the ambient noise user study involving the tasks of walking, text typing, and phone calling, we recruited twelve participants primarily from our university. The

average age of the participants was 35.42 (SD: 13.96) years, with a gender distribution of six females. Nine of the participants reported using their right hand as their dominant hand.

The duration of the user study for each participant was approximately twelve minutes.

To record ambient noise throughout the day, we invited five participants, with only one of them being affiliated with our university. The average age of these participants was 46.40 (SD: 16.24) years, with a gender distribution of two females. Three of the participants wore the smartwatch on their left hand. These participants had no prior knowledge of our research and were not associated with it. They also did not receive any compensation for their participation.

This user study spanned a minimum of eight hours.

We conducted a two-factor user study with a within-subjects design, focusing on mobile devices and four conditions. In the in-field user study, instead of counting swipe movements, we tracked the time spent on different activities. For each activity in the user study, such as walking, text typing, . . ., whole day, we counted the number of extracted sliding windows, categorized by $l \in \{64, 128\}$ samples.

So far, we have discussed the three user studies that we conducted. In the laboratory user study, we identified the best and worst conditions for the classification of the different swipe surfaces based on different swipe movement behaviors. One of these conditions involved creating an SVM model for both mobile devices, while the other conditions involved creating separate SVM models for smartphones and smartwatches. These conditions were detailed in Table 5. We evaluated the six different SVM models listed in this table with respect to the whole swipe from Section 6.1, in-field user study from Section 6.2, and ambient noise from Section 6.3. The following section presents the results of these user studies.

7. Results

We compare the results of human-determined and software-determined aspects. In this section, human-determined aspects refer to how users swipe, while software-determined aspects involve different SVM models and the accuracy of classifying swipe surfaces.

7.1. Human-Determined Aspects

First, we discuss the questionnaire from the laboratory user study regarding the swipe behaviors over the swipe surfaces. The obtained answers are illustrated in Tables 7 and 8. These two tables should give a first indication in which direction it might be possible to go.

We start by explaining the result in Table 7. In this table, the column *Filled in* shows the result of the questionnaire in the laboratory user study. The column *Applied* lists the selected swiping behavior in the field user study. This table compares the suggestions for human-determined aspects in the laboratory user study questionnaire and their aspects in the field user study. Our interpretation of the result in this table is that users should not have to choose between different swipe contacts. We noticed that some of the participants in the user studies had very short nails, which made the nail swipe contact irrelevant for them. In addition, one participant had artificial fingernails, which initially seemed suitable for the nail swipe contact. In practice, when the artificial nails were broken or protected by the user, it became difficult to perform the swipe contact correctly. In such cases, it was impossible to distinguish between the nail and skin swipe contacts. We observed that the questionnaire responses were generally consistent with the actual findings.

Table 7. Preferred swipe behaviors of the participants based on the questionnaire and the selected swipe behaviors in the field user study. In the column *Filled in*, we have no values in the rows *Dispreferred*, because the participants were free to choose their swipe behaviors. Therefore, we can assume that the participants preferred the swipe behaviors they chose.

	Swipe Behavior	Condition	Filled in in %	Applied in %
Preferred	Orientation	Horizontal	41.67	50.00
		Vertical	58.33	50.00
	Contact	Skin	75.00	75.00
		Nail	25.00	25.00
	Movement	Hand	100.00	100.00
		Finger	00.00	00.00
Dispreferred	Orientation	Horizontal	50.00	
		Vertical	50.00	
	Contact	Skin	16.67	
		Nail	83.33	
	Movement	Hand	25.00	
		Finger	75.00	

Table 8 lists the preferred and dispreferred swipe surfaces. The most disliked swipe surfaces were the breadbasket and splayed fingers, likely due to their irregular structure, which participants found unfavorable.

Table 8. Participants' preferred and dispreferred swipe surfaces based on the questionnaire in the laboratory user study.

	Preferred in %	Dispreferred in %
Small comb	41.67	8.33
Breadbasket	33.33	41.67
Notebook	16.67	8.33
Closed fingers	8.33	0.00
Splayed fingers	8.33	41.67

We also asked participants in the laboratory user study about their preferences regarding the size of the smartphone used for swiping over the surfaces. The evaluation of their responses revealed that none of the participants preferred a larger smartphone for this user study. Only 8.33% of the participants were satisfied with the given touchscreen size, while the majority preferred a smaller smartphone. The preference for a smaller smartphone is confirmed in the field user study, as participants exclusively used the *LG P700* smartphone, which is smaller than the smartphone used in the laboratory user study.

The results of the questionnaire must be interpreted with caution due to the limited sample size, although a trend is indicated, given the extremely small number of participants.

7.2. Software-Determined Aspect (Classification)

We outline the classification accuracy for the different swipe surfaces under different conditions as listed in Table 5. In this table, we listed the best and the worst conditions for the smartphone, the smartwatch, and both devices. We list the results for the laboratory user study, using the whole swipe movement in the laboratory user study and in the field user study.

Kostakos and Musolesi [86] argued that evaluating the SVM based solely on classifier accuracy is insufficient. Therefore, we present the data from various user studies in the form of a confusion matrix, also known as a confusion table or contingency table [40]. According to Kostakos and Musolesi [86], the presence of false positives is an important aspect that is often overlooked in study evaluations. By using a confusion matrix, we can

consider false positives and shed light on this aspect. The feature vectors were created using data from the laboratory, field, and ambient noise creation tasks. Figure 6 depicts the confusion matrix for the best conditions, as listed in Table 5. Figure 7 illustrates the confusion matrix for the worst conditions, also from the same table.

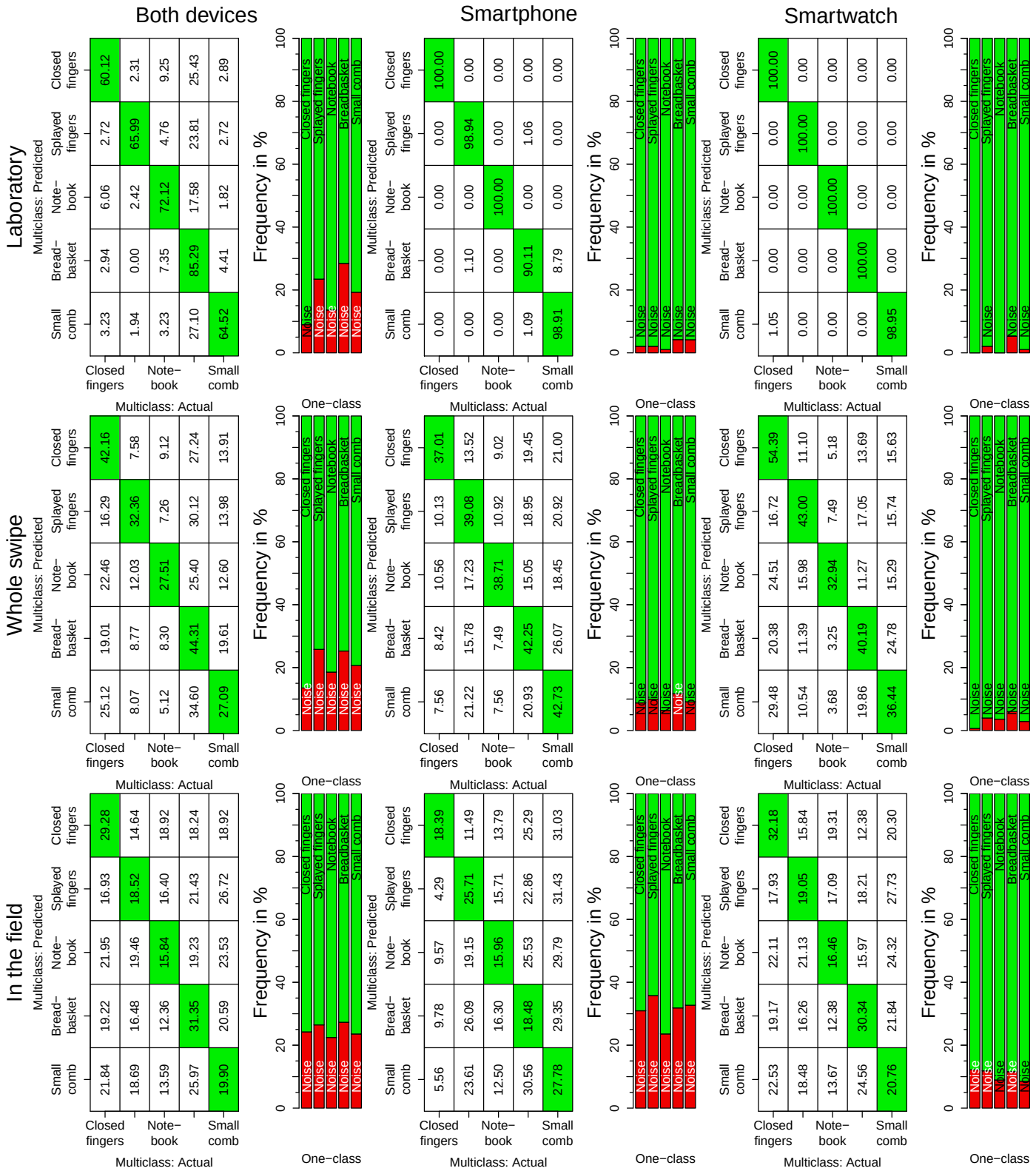


Figure 6. Confusion matrices (multiclass SVMs) and bar charts (one-class SVMs) for the best conditions from Table 5 are indicated by the column name. The row names represent the user studies.

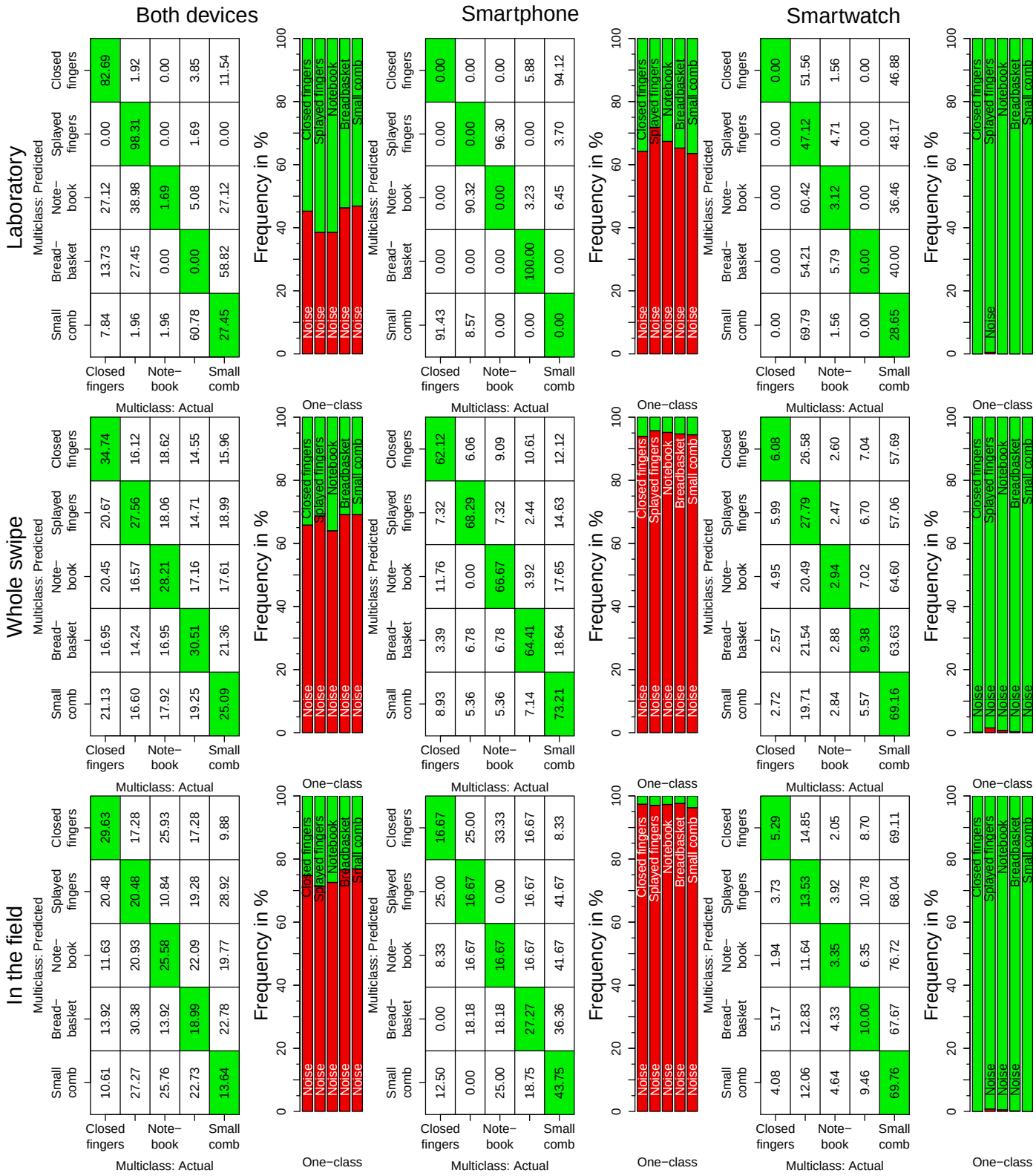


Figure 7. Confusion matrices (multiclass SVMs) and bar charts (one-class SVMs) for the worst conditions from Table 5 are indicated by the column name. The row names represent the user studies.

The structure of these figures resembles a table, with columns indicating the used SVM kernel and rows highlighting the data submitted to the SVM. The name of the column in these figure list the SVM model from Table 5. For example, the row labeled *Laboratory* represents data from the laboratory user study. All the displayed confusion matrices

show classification accuracy in percentages. The one-class SVM, as described in Figure 2, classified the ambient noise successfully. The bar chart to the left of the confusion matrix illustrates the classification success visually.

Upon analyzing the presented confusion matrices, we observed that SVMs performed exceptionally well under the best conditions, specifically in the laboratory. This observation holds true for smartphones, smartwatches, or a combination of both. However, in the worst SVM conditions, the classification accuracy significantly decreased compared with the best conditions. Under the evaluation conditions of whole swipe and in the field, the accuracy of swipe surface classification declined compared with the laboratory condition. This decline was observed irrespective of whether we considered the best or worst SVM conditions. Furthermore, regardless of the condition, the one-class SVM rejected approximately half of the extracted sliding windows from a single swipe movement. This finding is reflected in the confusion matrix for both the whole swipe and in the field conditions.

Ruuska et al. [87] emphasize that validation methods are not always in agreement. Therefore, they recommend using several methods for validation. Ben-David [88] concludes that simply counting the number of misses for each swipe surface, as in the confusion matrix, may be misleading when assessing the accuracy of swipe surfaces. Hence, Table 9 lists Cohen's κ for these confusion matrices as an additional validation method. Other most common statistical methods or metrics are, for instance, the coefficient of variation and Kendall's coefficient of concordance [89]. We chose to use Cohen's κ coefficient because it is generally considered a more reliable measure than a simple percent agreement calculation. It is because it takes into account the agreement occurring by chance [90]. We applied the interpretation of Cohen's κ according to McHugh [91]. κ can range from -1 to $+1$. A negative κ represents an agreement worse than expected. Low or negative values for κ represent no agreement and lead to the same result as in the confusion matrices.

Table 9. Calculation of Cohen's κ for the best and worst conditions from the confusion matrix in Figures 6 and 7. Thus, κ is calculated only for the multiclass SVM. The range of κ is computed with a 95% confidential interval (CI).

		Best Condition				Worst Condition			
User Study	**Device**	κ	κ-CI		**Agreement**	κ	κ-CI		**Agreement**
Laboratory	both devices	0.615	0.575	0.655	moderate	0.284	0.220	0.348	minimal
Laboratory	smartphone	0.970	0.953	0.988	almost perfect	0.007	−0.069	0.082	none
Laboratory	smartwatch	0.997	0.992	1.000	almost perfect	−0.053	−0.078	−0.028	none
Whole swipe	both devices	0.184	0.171	0.197	none	0.116	0.096	0.137	none
Whole swipe	smartphone	0.250	0.231	0.270	minimal	0.581	0.511	0.652	weak
Whole swipe	smartwatch	0.268	0.251	0.285	minimal	0.036	0.028	0.045	none
In the field	both devices	0.039	0.016	0.061	none	0.025	−0.026	0.076	none
In the field	smartphone	0.014	−0.034	0.063	none	0.056	−0.075	0.188	none
In the field	smartwatch	0.048	0.025	0.072	none	0.005	−0.009	0.019	none

In Figures 8 and 9, we aim to highlight the impact of swiping while sitting and standing, as shown in Figure 5, as well as the hand pose, as depicted in Table 6. These are the findings from the user study conducted in the field, as discussed in Section 6.2.

These figures follow the same format as the previous two. If a row is labeled *One device*, it indicates no connection between the smartphone and the smartwatch, while the row labeled *Both Devices* signifies a Bluetooth connection between the two mobile devices. If we used a Bluetooth connection in the in-field user study, then the participants used the mobile devices at the same time as was illustrated in Table 6. Without the Bluetooth connection, the hand pose changed, as was illustrated also in this table. In the laboratory user study, we also had a Bluetooth connection. However, in that study, we did not differentiate the swipe orientation, as depicted in Figure 1.

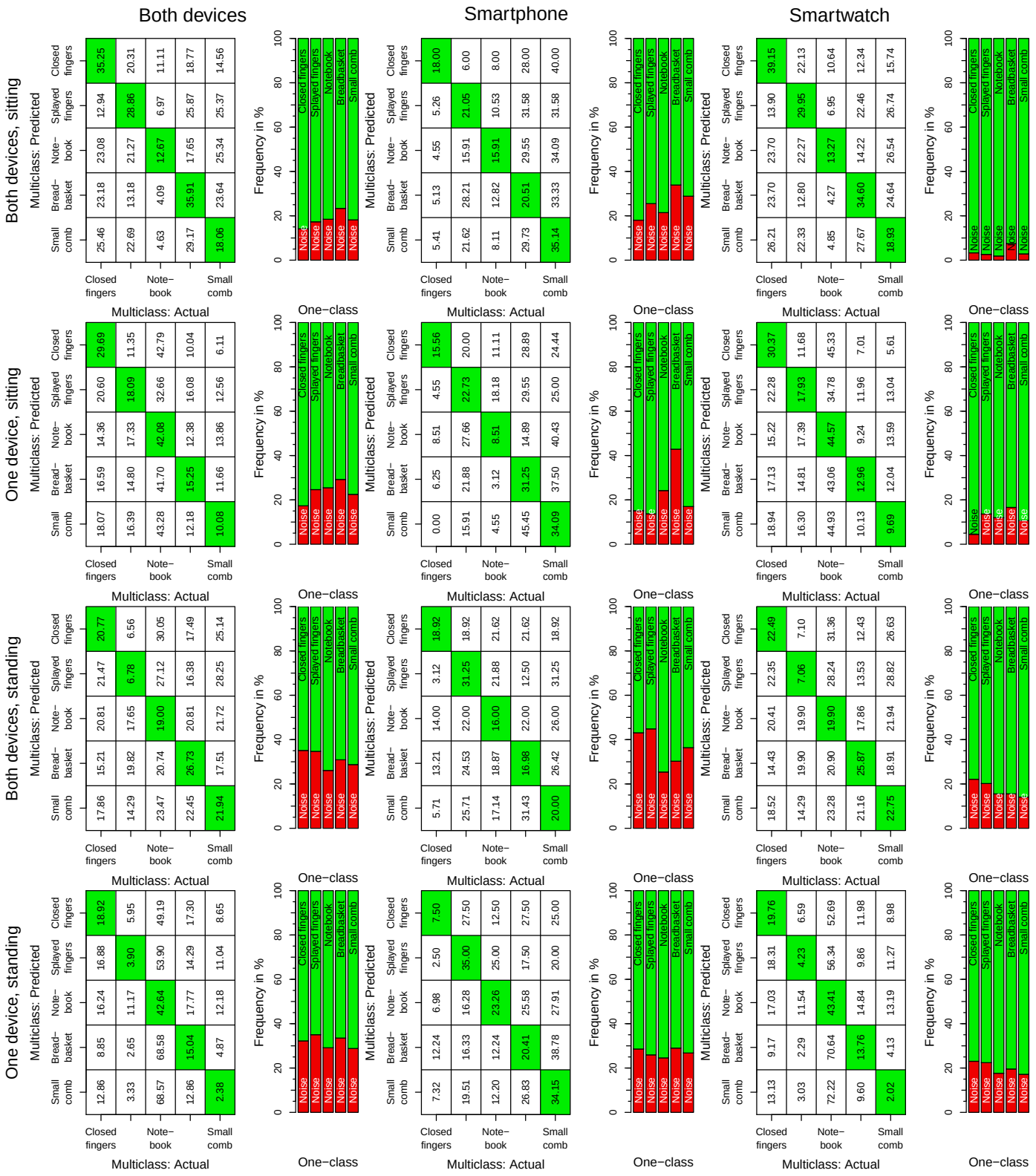


Figure 8. Confusion matrices (multiclass SVM) and bar charts (one-class SVM) illustrating the results under the best conditions during the in-field user study. The column name in the matrices represents the best conditions for the SVM, which was listed in Table 5. The row name in the matrices corresponds to different hand poses, as depicted in Figure 6.

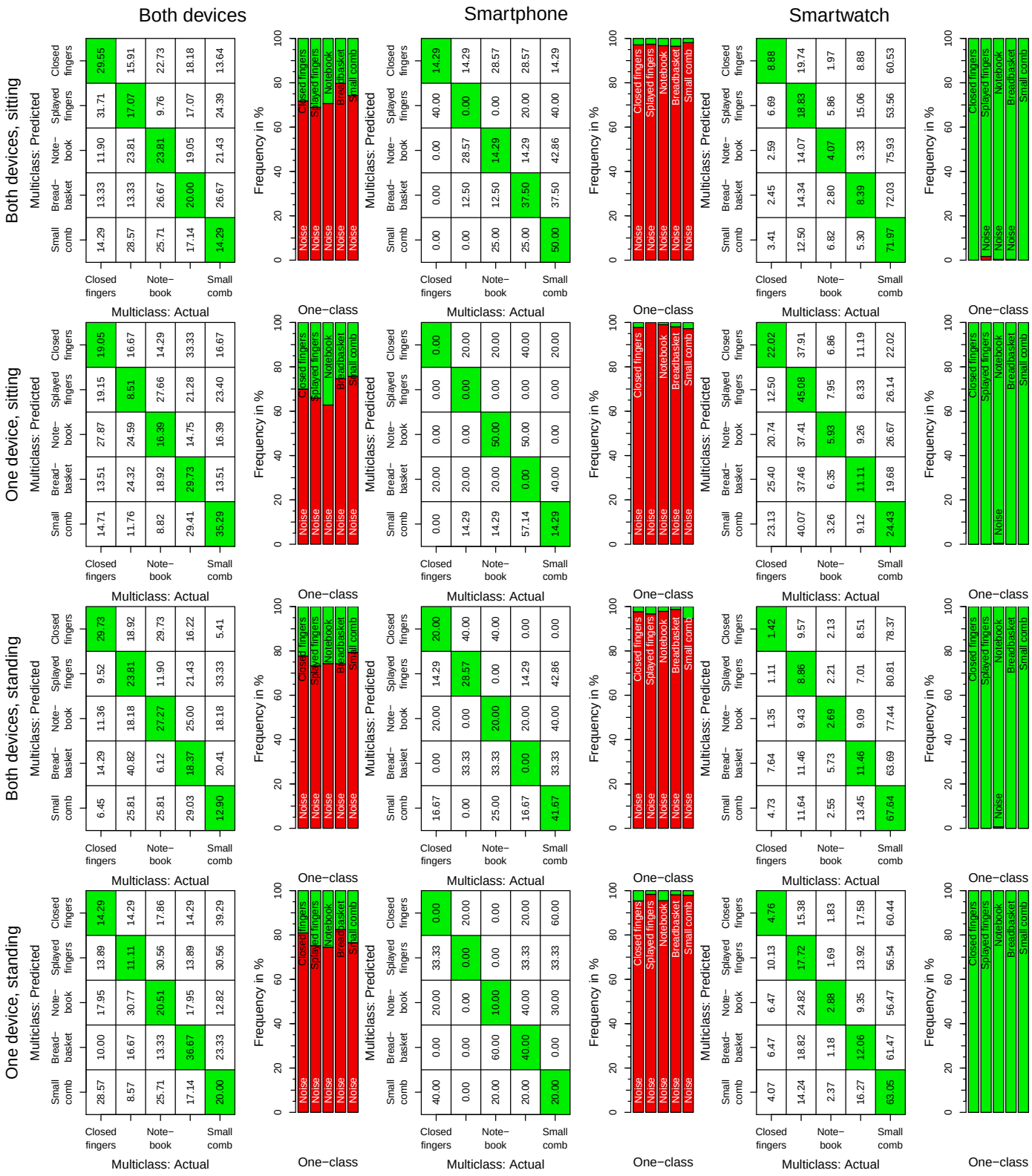


Figure 9. Confusion matrices (multiclass SVM) and bar charts (one-class SVM) illustrating the results under the worst conditions during the in-field user study. The column name in the matrices represents the best conditions for the SVM, which was listed in Table 5. The row name in the matrices corresponds to different hand poses, as depicted in Figure 6.

The shown confusion matrices and bar charts reveal a distinction based on whether the smartphone is held in the hand or not. However, we did not explore the specific reasons for this difference in detail. Further investigation into this matter is beyond the scope of this work, and therefore, we cannot provide any additional interpretations of the presented data.

It is worth noting that all participants preferred using the smaller smartphone, as was used in the laboratory study.

We calculated also Cohen's κ only for the confusion matrices of the multiclass SVMs. The result of this calculation is listed in Table 10. Also in this case, Cohen's κ leads to the same interpretation as the confusion matrices in Figures 8 and 9.

Table 10. Calculation of Cohen's κ for the best and worst conditions from the confusion matrix in Figures 8 and 9. Thus, κ is calculated only for the multiclass SVM. The range of κ is computed with a 95% confidential interval (CI).

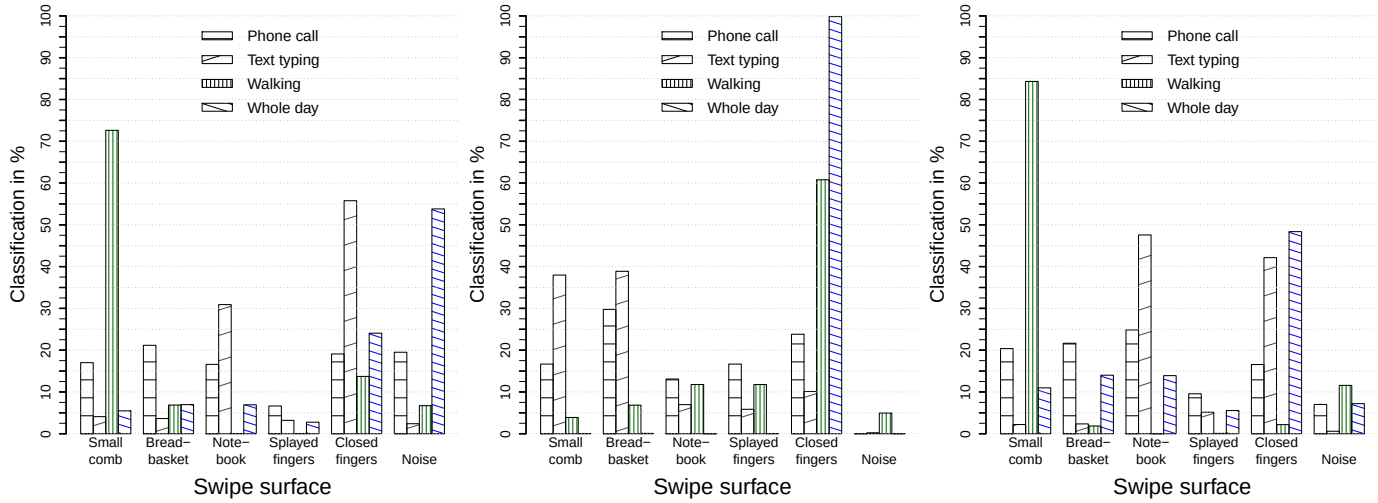
			Best Condition				**Worst Condition**			
Device	**Used**		κ	κ-**CI**		**Agreement**	κ	κ-**CI**		**Agreement**
Both devices	both	standing	−0.010	−0.041	0.020	none	0.030	−0.043	0.103	none
Both devices	both	sitting	0.079	0.047	0.111	none	0.015	−0.057	0.087	none
Both devices	one	standing	−0.044	−0.072	−0.017	none	0.001	−0.075	0.077	none
Both devices	one	sitting	0.038	0.008	0.068	none	0.009	−0.057	0.075	none
Smartphone	both	standing	0.000	−0.068	0.067	none	0.062	−0.124	0.249	none
Smartphone	both	sitting	0.032	−0.036	0.100	none	0.042	−0.126	0.211	none
Smartphone	one	standing	0.048	−0.023	0.118	none	−0.090	−0.229	0.049	none
Smartphone	one	sitting	0.030	−0.036	0.097	none	−0.133	−0.336	0.070	none
Smartwatch	both	standing	−0.003	−0.035	0.029	none	−0.018	−0.037	0.001	none
Smartwatch	both	sitting	0.092	0.059	0.125	none	0.030	0.008	0.052	none
Smartwatch	one	standing	−0.043	−0.072	−0.014	none	0.001	−0.021	0.023	none
Smartwatch	one	sitting	0.039	0.008	0.069	none	0.021	−0.004	0.046	none

In Appendix E, we explain the process of calculation of κ.

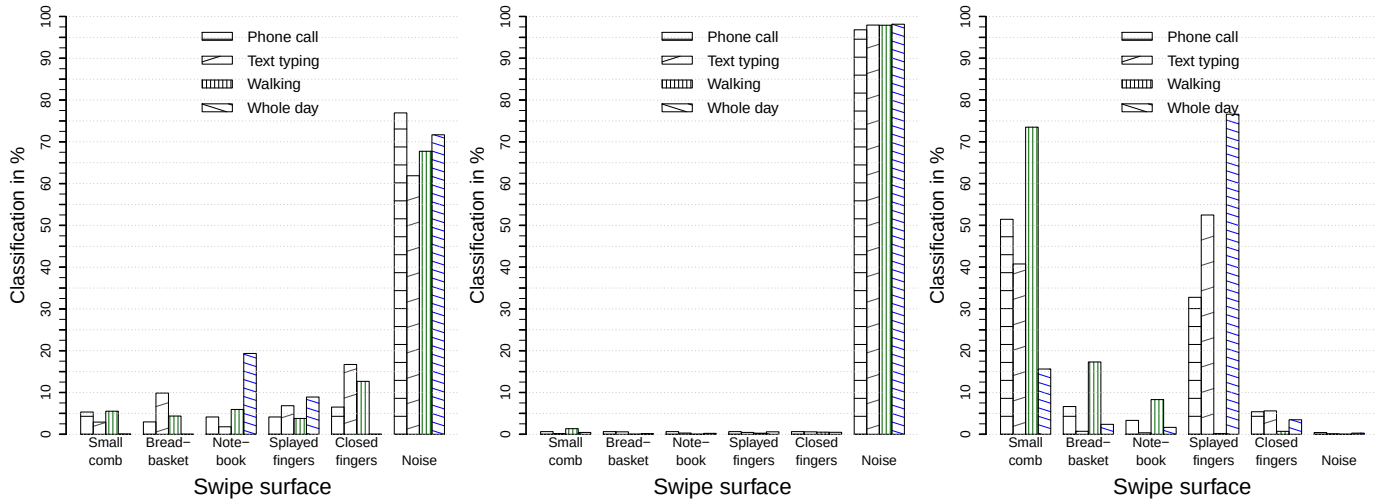
Figure 10 showcases the various tasks that generated ambient noise signals. In the optimal condition, the SVM classifies 100% of the recorded human body activities as ambient noise. We achieve a near-optimal case with the SVM kernel for the smartphone, as shown in Figure 10b. However, in the remaining cases, the SVM struggles to distinguish between a potential swipe surface and ambient noise.

In Appendix F, we outline also the shape of the feature vector $\vec{Y}$.

In the previous section, we compared the answers from the questionnaire with the actions in the field user study, where we found that the results were similar. We then discussed the evaluation of SVM-based classification for swipe surface recognition, presenting data from user studies conducted in different environments. We also analyzed the shapes of the resulting feature vectors. We discuss our findings in the next section.



(a) Best conditions



(b) Worst conditions

Figure 10. Misclassified ambient noise signals as swipe gestures. The best and the worst conditions are taken from Table 5.

8. Discussion

We compare our outcome with the existing literature, exploring both human- and software-related aspects.

8.1. Human-Determined Aspects

The role of human-determined aspects in achieving high accuracy in swipe surface classification is significant. When these aspects are not adequately trained, it leads to a drastic decline in classification accuracy. Previous research by Han et al. [2] proposed a gesture involving the use of a fingernail, which has been found to be unfavorable for this interface. User studies have revealed that the availability of a required fingernail is not always guaranteed. On the other hand, the horizontal and vertical swipe orientations suggested by Han et al. [2] have shown promise. However, we concur with the findings of Han et al. [2] that an individual's swiping styles may vary from swipe to swipe. These variations in swiping style must be taken into account for this swipe-based interaction.

We have adopted and extended the swipe movement behaviors over textured surfaces as outlined by Han et al. [2], considering that users hold smartphones in their hands. Table 1 lists these behaviors. Additionally, the gestures described in Table 6 offer a new perspective on swipe surface interaction. However, it is important to note that while

choosing an appropriate gesture is crucial, placing emphasis on the software aspects is even more critical. In the following section, we discuss these software aspects in relation to the existing literature.

8.2. Software-Determined Aspects

We conducted experiments to confirm the sampling frequencies of the smartphone and smartwatch. The smartphone operates at an approximate sampling frequency of 100 Hz, while the smartwatch has a higher sampling frequency. Similar to the findings of Lukowicz et al. [17] and Owusu et al. [34], we were able to achieve accurate classification of different swipe surfaces using these sampling frequencies. We also acknowledge Han et al.'s [2] concern regarding the lower sampling rate of accelerometers in mobile devices compared with acoustic sensors, which can affect the sensitivity in detecting vibration signals.

Through our investigation, we found that the most successful swipe surface classification was achieved when using a feature vector consisting of samples in the frequency domain, such as mels. We observed that additional signal descriptors were not necessary for the frequency domain samples. Analyzing the feature vector plots provided us with a better understanding of its elements. We noticed that the feature vector performed exceptionally well under laboratory conditions, but its classification accuracy significantly decreased during the evaluation. This aligns with the viewpoint expressed by Saeys et al. [39] that deeper insights into the underlying data generation processes are necessary to achieve high swipe surface classification accuracy. We also concur with Sherin and Supriya [38] on the importance of selecting an appropriate feature vector, as it directly contributes to achieving high classification accuracy. These considerations pertain to the interpretability and explainability of feature vectors and the classification of different swipe surfaces [92,93]. Interpretability primarily relates to the intuition behind the outputs of the SVM kernel [93], while explainability focuses on the internal logic and mechanics within an SVM [93]. Ribeiro et al. [92] emphasized that accuracy often proves to be an unsuitable metric for evaluating SVM kernels, motivating the need for explaining SVM kernels [92]. However, Linardatos et al. [93] found a lack of mathematical formality and rigor in both interpretability and explainability, despite several attempts made. We concur with [22], who observed that the selection of elements in the feature vector remains an ongoing research topic in supervised learning, closely tied to feature vector engineering [92]. Two types of models need to be considered for these concepts. An SVM kernel represents one such model. We employed SVM kernels as black-box models. Conversely, we have the so-called white-box or glass-box models. White-box models yield explainable results at the cost of reduced power, failing to achieve state-of-the-art performance compared with black-box models [93].

Our findings demonstrate that SVM is capable of effectively separating ambient noise from potential swipe surfaces, indicating that training an additional class is unnecessary. However, we observed that the classification accuracy can be further improved. Under this assumption, we anticipate better classification results for different swipe surfaces compared with the current performance.

The presence of a first-order Butterworth high-pass filter has an impact, suggesting the necessity of a filter. We believe that further investigation is required to determine the specific details of ω_c in order to make definitive statements.

For $l \in \{64, 128\}$ samples with $f_s \approx 100$ Hz, it is sufficient to classify different swipe surfaces. However, a higher sample rate might lead to improved classification accuracy. It is worth noting that a constant sample rate, rather than a higher one, might be the crucial factor. We discovered variations in the sample frequency, causing the feature vector to differ due to missing samples required for its calculation. Consequently, the feature vector contains elements that describe another swipe surface, leading to the misclassification of swipe surfaces.

The RBF kernel and a linear kernel demonstrate high classification accuracy and serve as a solid starting point for further research. Lin and Lin [94] emphasized that the RBF kernel should be the first choice. However, to effectively classify swipe surfaces using these kernels, appropriate hyperparameters need to be determined. The default values for the hyperparameters result in low classification accuracy for swipe surfaces. Additionally, Lin and Lin [94] indicated that under certain parameter settings, the sigmoid kernel behaves similarly to the RBF kernel, while the linear kernel can be seen as a special case of the RBF kernel among existing SVM kernels.

We agree with Manevitz and Yousef [60] that $\vec{Y}$ should consist of a small number of elements. Specifically, for the RBF SVM kernel, Manevitz and Yousef [60] demonstrated that 10 elements in $\vec{Y}$ yield significantly better results than 20 elements. However, it should be noted that generalizing these findings to other applications can be challenging.

9. Conclusions

In this work, we have addressed additional considerations that were not explored by Harrison et al. [4] and Han et al. [2]. We considered human-determined aspects and software-determined aspects. The human-determined aspects consider the manner in which users swipe over the swipe surfaces and the location of the mobile devices. The software-determined aspects pertain to the length of the sliding window w, the cut-off frequency w_c for the Butterworth high-pass filter, the kernels for the one-class SVM and the multiclass SVM, and the feature elements in the feature vector $\vec{Y}$. In particular, the software-determined aspects have been included in the flow diagram. Figure 2 shows this flow diagram, which can be used to detect swiping gestures on textured surfaces. However, the results of our conducted user studies indicate that users tend to perform the gestures imprecisely when using this interface. This imprecision can be attributed to factors such as the size of the smartphone being too large for comfortable handling or the fingernails being cut too short to make effective contact with the swipe surface. Nevertheless, considering this "sloppy" execution of gestures, our findings clearly highlight the need for future research to focus on software-determined aspects that can accommodate moderately imprecise swipe gestures in this interface.

The research question can be answered by identifying the key software-determined aspects that influence the vibration-based input on mobile devices for recognizing various textured surfaces using SVM. Our work also provides guidance on the specific software-determined aspects that require special attention to achieve accurate classification of textured surfaces. In the following, we briefly discuss these software-determined aspects.

In the laboratory user study, we achieved high surface classification using the mels from MFCC approach (see Equation (11) on page 18), which aligns with research on rapid recognition of speed [65]. Gaspar et al. [65] emphasized the importance of selecting the appropriate kernel, tuning hyperparameters, adjusting the misclassification penalty C, and carefully choosing the elements for the feature vector. Additionally, a suitable filter for the vibration signal must be selected to attain high classification accuracy. It is crucial to keep the feature vector concise, as an increase in the number of elements in $\vec{Y}$ significantly decreased the classification performance of different swipe surfaces [60].

We acknowledge that our study has limitations because we only used textured surfaces. We assumed that the SVM could easily distinguish vibration signals from these surfaces. It's important to first confirm if our findings can apply to vibration signals of surfaces with small amplitude differences. Although other machine learning algorithms like neural networks could also be used, we chose to stick with the SVM for detecting the vibration signals of different surfaces, inspired by Han et al. [2]. Recognizing this limitation, we understand the need for further research on the human-perceived aspects of this vibration-based interface. In the next section, we provide a list of additional research topics to explore in future investigations.

9.1. Outlook

9.1.1. Human-Determined Aspects

In this study, we demonstrated that participants have a preference for smaller smartwatches. However, there is a need for further research to investigate how smartphones can be effectively held in the hand. Currently, it remains unclear how handles designed for smartphones can enhance the usability of larger devices and how they may impact the detection of vibration signals. Conducting research in this area will provide valuable insights into the ergonomic considerations and potential implications for vibration signal detection.

9.1.2. Software-Determined Aspects

Suitable values for hyperparameters: One approach to address the optimization problem is the Bat Algorithm [38,95]. This algorithm is a metaheuristic method inspired by the echolocation behavior of bats [95]. Another technique that can be employed is grid search [49,62,63], along with other similar algorithmic methods [96]. These approaches offer potential strategies to find suitable hyperparameters for SVM kernels and enhance the accurate classification of various swipe surfaces.

Sample rate: The sample rate on mobile devices running the *Android* operating system often varies, leading to lower classification accuracy for different swipe surfaces. To address this issue, Liu et al. [33] employed cubic spline interpolation, a widely used signal-processing technique, to correct nonuniformly sampled data. An alternative approach involves utilizing the native language C/C++ for Android, which offers a nearly constant sampling rate of 200 Hz [97]. The consideration of sample rate arose due to the use of a commercially available, low-cost vibration sensor by Han et al. [2], as such sensors typically maintain a consistently high sample rate. By addressing the variations in sample rate, we can improve the reliability and accuracy of the classification process for different swipe surfaces.

Vibration signal synchronization: We observed variations in hand movements, which is consistent with the findings by Guerra-Casanova et al. [98] and Liu et al. [33]. In order to enhance the accuracy of swipe surface classification in future work, it is crucial to synchronize these different human behaviors. We believe that synchronizing hand movements can improve classification accuracy. A potential way to achieve synchronization is by using dynamic time warping (DTW) [30,99] or its extension locally slope-based dynamic time warping (LSDTW) [100]. Furthermore, we need to determine whether a sample size of 64 or 128 is enough to capture the vibration signal from a swipe surface. Understanding the optimal sample size will help to improve the classification process.

Author Contributions: Conceptualization, T.H. and D.A.; methodology, T.H. and D.A.; software, T.H.; validation, T.H., D.A. and M.H.; formal analysis, T.H.; investigation, T.H.; resources, T.H.; data curation, T.H.; writing—original draft preparation, T.H.; writing—review and editing, T.H., D.A. and M.H.; visualization, T.H.; supervision, M.H.; project administration, T.H., D.A. and M.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Study was exempt from referral to the University's ethics committee. Austrian legislation does not mandate establishing an ethics committee unless the university in question is a medical university (cf. Section 30 of the University Act 2002). Additionally, the study performed poses minimal risk to human subjects and are not eligible for ethical approval (45 C.F.R. § 46.102(i); Code of Federal Regulations, USA).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study. However, the identity of the participants remains unknown.

Data Availability Statement: The experimental data of the studies described in this paper will be made available on the research team's homepage https://www.aau.at/en/isys/ias/research/ upon publication.

Acknowledgments: We would like to thank the reviewers and the editorial team for their valuable comments and suggestions. We would also like to thank all study participants for their contributions, which made the user studies possible.

Conflicts of Interest: The authors declare no conflicts of interest. Participants in user studies described in this work were not compensated for their involvement.

Appendix A. Swipe Gesture

The term *swipe gesture* refers to the same action of moving one's finger across a surface in a single continuous motion. The word *swipe gesture* is commonly used and recognized in technical contexts, especially when discussing interactions with digital devices or user interfaces.

In addition, we use the word *swipe* to indicate a single instance of the action, while *swiping* refers to the ongoing or repeated action.

Appendix B. Single-Board Computer

The Odroid N2+ CPU operates at a maximum clock frequency of 2.4 GHz on quad-core Cortex-A73 and 2.0 GHz on dual-core Cortex-A53. To maintain high clock frequencies for extended periods, a fan was utilized. Without the fan, the computer would heat up, resulting in decreased clock frequency and longer computation times. We employed *GraalVM*, see more details on the web page of GraalVM: https://www.graalvm.org/, last access 15 October 2022, version 22.2.0 for the *AArch64* chip architecture (community edition) to enhance the performance of the Java code by generating native binaries. The Odroid N2+ is equipped with 4 GB of memory, and to avoid exceeding the memory limit, we trained only four different SVMs simultaneously.

Appendix C. Vibration Signal

Figure A1 demonstrates the effect of different swipe movement behaviors on the recorded vibration signal from the smartphone and smartwatch separately. In this figure, the three acceleration directions x, y, and z are depicted. These plots are aligned with the representation of the sliding window w in a matrix structure, as described by Equation (1).

In this figure, the closed fingers symbolize the swipe surface. These two figures highlight the varying swipe behaviors that result in different vibration signal amplitudes. These distinct amplitudes should contribute to different elements in the feature vector $\vec{Y}$, thereby facilitating the accurate classification of swipe surfaces.

In Figure A1, the recorded vibration signal with the finger and hand is displayed. The time period of 1.25 to 3.25 s was utilized for w_i, resulting in a duration of 2 s denoted as T. The sampling frequency f_s on mobile devices is approximately 100 Hz. The total number of samples within this time frame is represented by N, calculated as $N = f_s \cdot T = 100\ \text{Hz} \cdot 2\ \text{s} = 200$ samples. To account for the half-overlap used, the number of samples should be greater or nearly equal to 192 samples. We assume a maximum of 128 samples within w_i due to the half-overlap and the train/test split method, which requires 3 sets of 64 samples. This satisfies the condition of extracting three ws from a swipe movement. However, extracting more than three w_i per swipe movement would increase the likelihood of including samples with ambient noise information. If we cannot extract at least three w_i, the swipe movement is not considered for training the SVM.

In the previous determination of the required samples, we assumed 100 Hz. Now, we want to determine the sampling frequency present on the two mobile devices. Therefore, we used the recorded vibration signals from the user study. For this, we used the number of samples N within a swipe movement gesture over a surface. The time required to record the samples of this swipe gesture was also measured and denoted as Δt with second as unit. Then we can determine the sampling frequency f_s with $N/\Delta t$. We obtained for the smartphone an overall measured sampling frequency of 102.15 Hz (SD: 0.91 Hz). The smartwatch had an f_s of 124.42 Hz (SD: 6.65 Hz). From the calculation of sampling frequency f_s, we saw that the f_s on the smartphone was nearly constant. On the smartwatch,

the sampling frequency varies more than on the smartphone. We have enough samples N so that we can extract three different sliding windows from each swipe movement. The smartphone recorded on average 658.22 (SD: 245.71) samples. The other device recorded on average 795.01 (SD: 284.92) samples. On average, the swipe movement took 3.22 (SD: 1.21) seconds over each swipe surface on the smartphone. The samples were recorded for an average of 3.2 (SD: 1.17) seconds on the smartwatch. However, we have to consider that the hand moves to the swipe surface and releases the swipe surface after finishing the swipe movement. Thus, not every extracted sliding window contained information about a swipe surface.

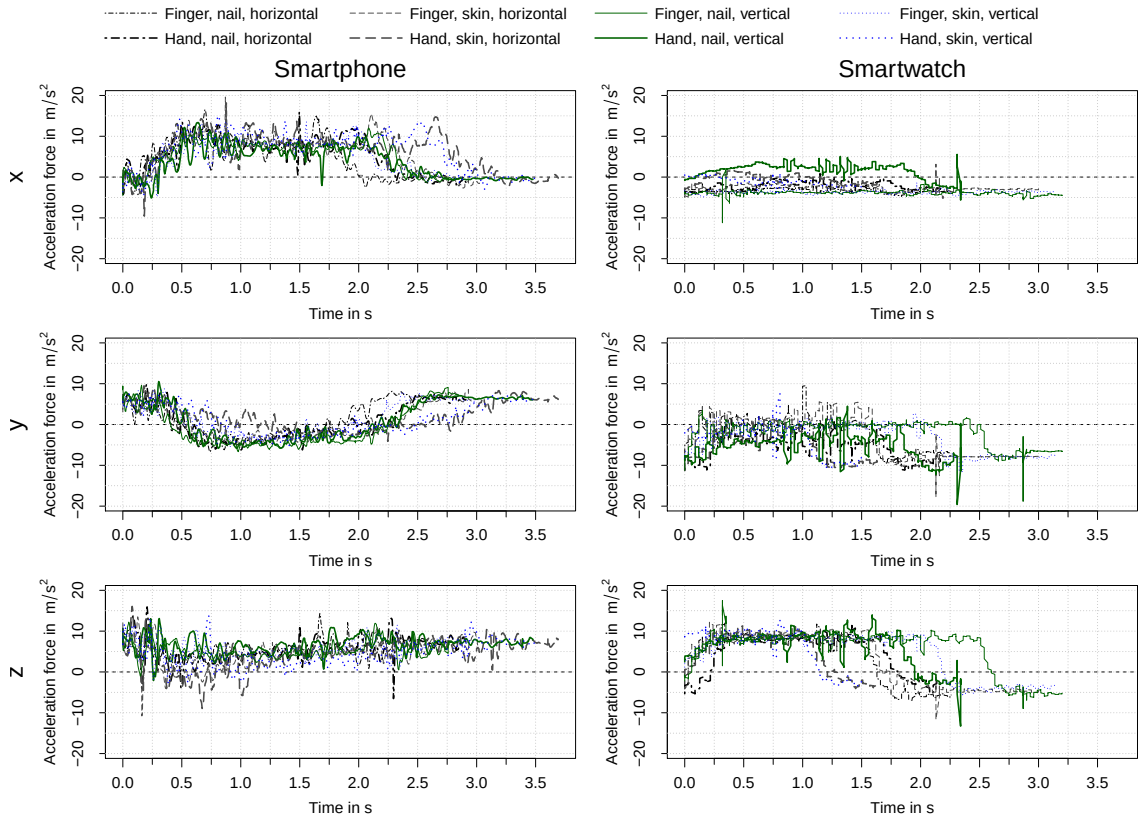


Figure A1. The vibration signal shape while participants swipe over their closed fingers.

Appendix D. D-Optimal Design

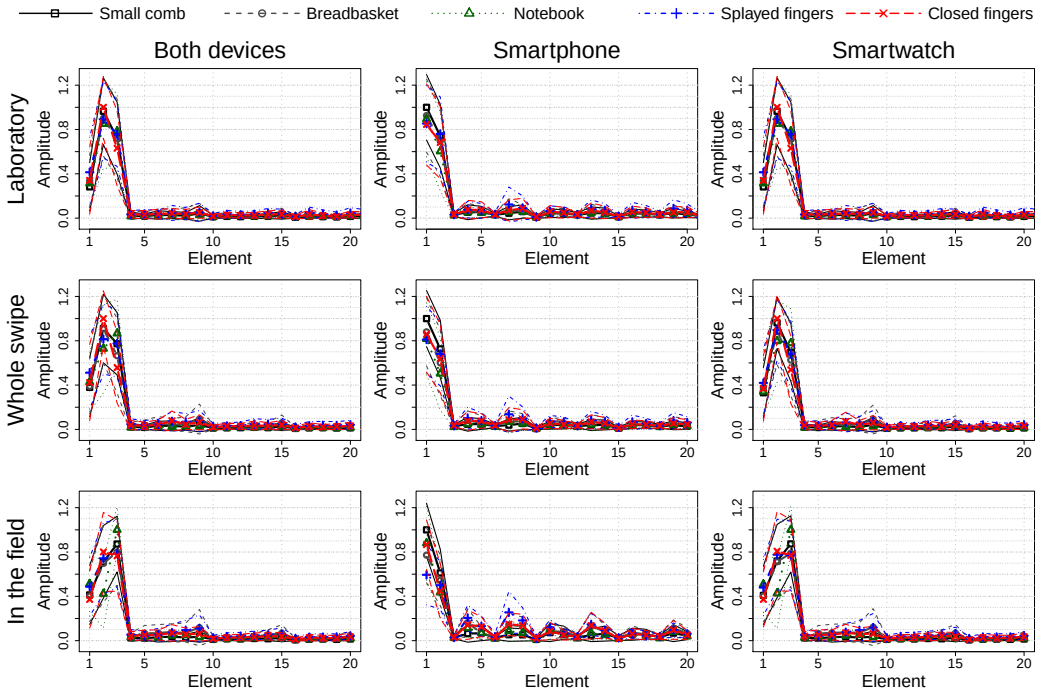
To implement the D-optimal design, we utilized the statistical software program *R*. In *R*, we used the `optFederov` command from the `AlgDesign` library. We set the parameters for `optFederov` as follows: `nTrails` = 300, `approximate` = `TRUE`, `criterion` = `D`, and `andevaluateI` = `TRUE`. By setting `nTrails` to 300, we reduced the full factorial design to the desired number of experiments. The `D` criterion was chosen to employ the standard approach of *D-efficiency* [101,102]. After applying the `optFederov` command with the specified parameters, we obtained the design, along with the values of the parameter D (0.199) and Ge (1.0). These values indicate that the selected parameters are appropriate, as D is low and Ge is close to one [103].

Appendix E. Cohen's Kappa

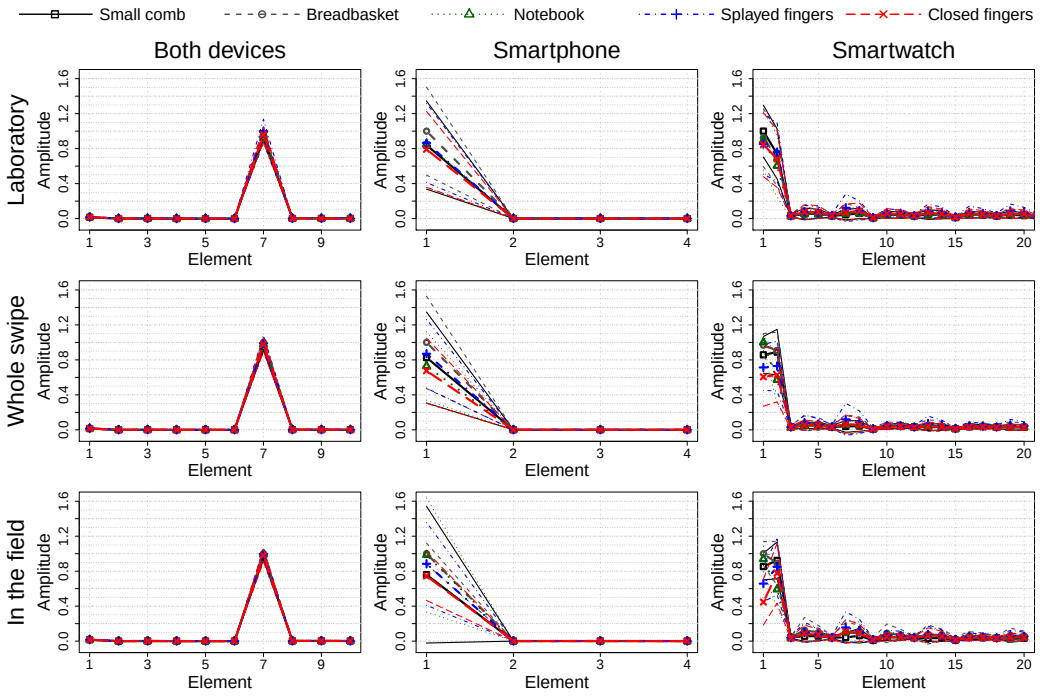
We used the statistical software program *R* to determine Cohen's κ coefficient. We used confusion matrices from Figures 6–9. Instead of calculating the percentage, we now used the count in each matrix. Then, we submit each matrix to the function `Kappa(...)` from the library *vcd*. After we executed the function `Kappa(...)`, we extract the result of κ from this function with the function `confint(...)`.

Appendix F. Shape of Feature Vector

To gain a deeper understanding of the feature vector, we discuss in this section the shape of the feature vector. To focus only on the elements in a feature vector where the classified wipe surface matches the given swipe surface, Figure A2 shows the values of the elements in feature vectors for the best and worst SVM conditions. These plots do not account for misclassified swipe surfaces or ambient noise.



(a) Best conditions

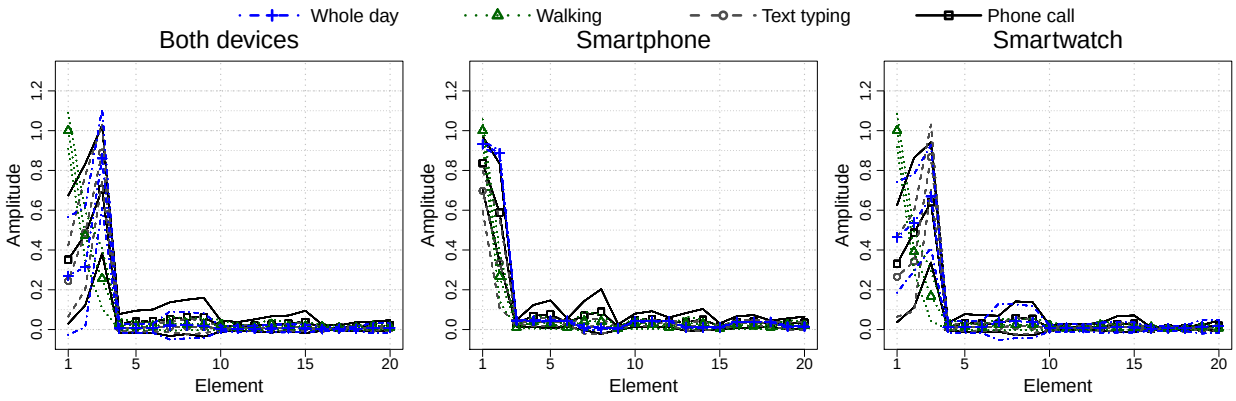


(b) Worst conditions

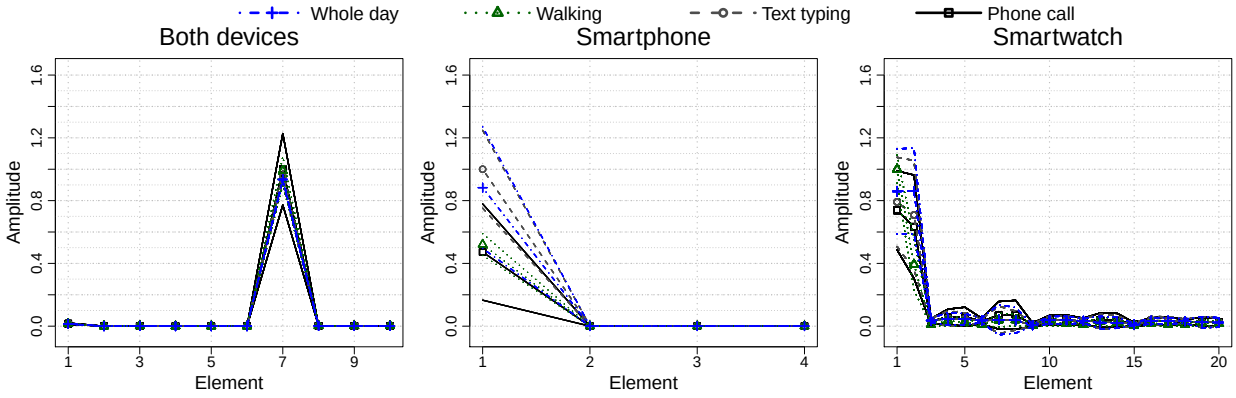
Figure A2. Feature vector shapes for swipe surfaces. The best and the worst conditions are taken from Table 5.

The y-axes in these plots have been normalized. To obtain the average and standard deviation (SD) of the feature vectors for each condition (e.g., the swipe surface *small comb*, the user study in the *laboratory*, and the usage of *both devices*), we calculated the average and SD of individual elements across all the generated feature vectors. This normalization allows us to compare the different values of feature elements in $\vec{Y}$. The plots only display the first 20 elements of each feature vector, even if they contain more elements. This displayed number of feature elements ensures that the feature vector can be properly represented in the plot. The number of elements in the feature vector were listed in Table 5. In these plots, the thicker line represents the mean values of each element in the feature vector, while the thinner line represents the standard deviation. The mean values of the elements are very close within the range of one to four, but the standard deviation is large in this range. Comparing the best and worst SVM conditions, the plots show that the feature vector elements are almost overlaid. Under these conditions, the swipe surface classification accuracy is extremely low, as also indicated by the confusion matrix in Figure 7. The overlapped SD of each element suggests a significant decline in classification accuracy for the whole swipe and in-field evaluation scenarios.

To examine the shape of the feature vector for ambient noise vibration signals, Figure A3 illustrates samples of w_i for the best and worst candidates under different mobile conditions.



(a) Best conditions



(b) Worst conditions

Figure A3. Feature vector shape for ambient noise signals. The best and the worst conditions are taken from Table 5.

Comparing the shape of $\vec{Y}$ in Figures A2 and A3, we see that the shape of the ambient noise feature vector is somewhat different from that of the vibration signal of the swipe surface. However, this difference in shape may be too small to significantly improve classi-

fication accuracy. This observation is supported by Poel [21], who noted that mislabeled data or noise in the labels can adversely affect the classification performance of SVMs.

Figure A4 visually illustrates the schema the theoretically worst and the best shape for the feature vector.

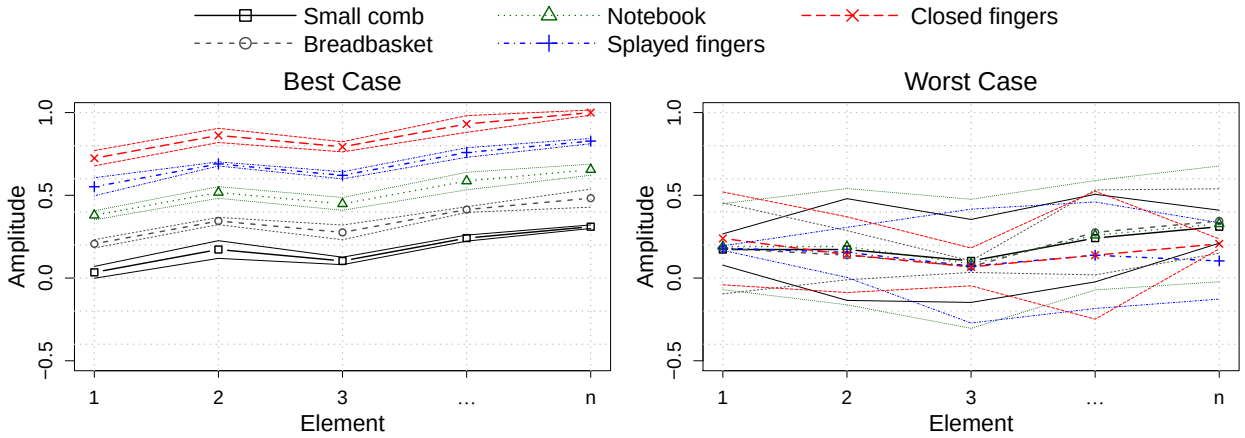


Figure A4. Schematic of the best and worst feature vector conditions for high classification accuracy of different swipe surfaces in Figure 1.

In this schematic sketch, the dotted line represents the mean amplitude value of an element in $\vec{Y}$, while the solid lines above and below it indicate the standard deviation (SD) of this amplitude value. This visual representation aims to illustrate the variations in the feature vector elements under different conditions. On the left sketch, we can observe the ideal shape of feature vector elements that indicate high classification potential. Another sketch showcases the actual form of elements obtained from user studies, where the SD closely aligns with the mean of each element in $\vec{Y}$. It is important to note that the means of the elements in the feature vector do not overlap and are not equal to their corresponding standard deviations, which contributes to achieving high classification accuracy.

References

1. Kratz, S.; Rohs, M.; Wolf, K.; Müller, J.; Wilhelm, M.; Johansson, C.; Tholander, J.; Laaksolahti, J. Body, Movement, Gesture & Tactility in Interaction with Mobile Devices. In Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services, New York, NY, USA, 30 August–2 September 2011; MobileHCI'11; pp. 757–759. [CrossRef]
2. Han, T.; Ahlström, D.; Yang, X.D.; Byagowi, A.; Irani, P. Exploring Design Factors for Transforming Passive Vibration Signals into Smartwear Interactions. In Proceedings of the 9th Nordic Conference on Human-Computer Interaction, New York, NY, USA, 23–27 October 2016; NordiCHI'16; pp. 35:1–35:10. [CrossRef]
3. Awada, A.; Issa, Y.B.; Tekli, J.; Chbeir, R. Evaluation of Touch Screen Vibration Accessibility for Blind Users. In Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility, New York, NY, USA, 21–23 October 2013; ASSETS'13; pp. 48:1–48:2. [CrossRef]
4. Harrison, C.; Xiao, R.; Hudson, S. Acoustic Barcodes: Passive, Durable and Inexpensive Notched Identification Tags. In Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology, New York, NY, USA, 7–10 October 2012; UIST'12; pp. 563–568. [CrossRef]
5. Dictionary.com. Swipe. Web Page. 2023. Available online: https://www.dictionary.com/browse/swipe (accessed on 24 May 2023).
6. Parera, J.; Angulo, C.; Rodríguez-Molinero, A.; Cabestany, J. User Daily Activity Classification from Accelerometry Using Feature Selection and SVM. In *Bio-Inspired Systems: Computational and Ambient Intelligence*; Cabestany, J., Sandoval, F., Prieto, A., Corchado, J.M., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1137–1144. [CrossRef]
7. Zhang, X.; Wu, G.; Dong, Z.; Crawford, C. Embedded feature-selection support vector machine for driving pattern recognition. *J. Frankl. Inst.* **2015**, *352*, 669–685. [CrossRef]
8. Pelletier, C.; Valero, S.; Inglada, J.; Champion, N.; Marais Sicre, C.; Dedieu, G. Effect of Training Class Label Noise on Classification Performances for Land Cover Mapping with Satellite Image Time Series. *Remote Sens.* **2017**, *9*, 173. [CrossRef]
9. Le, V.K.; Beauseroy, P.; Grall-Maes, E. Abnormal Trajectory Detection for Security Infrastructure. In Proceedings of the 2nd International Conference on Digital Signal Processing, New York, NY, USA, 25–27 February 2018; ICDSP 2018; pp. 1–5. [CrossRef]
10. Chang, C.C.; Lin, C.J. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.* **2011**, *2*, 27:1–27:27. [CrossRef]

11. John, G.H.; Kohavi, R.; Pfleger, K. Irrelevant Features and the Subset Selection Problem. In *Machine Learning Proceedings 1994*; Cohen, W.W., Hirsh, H., Eds.; Morgan Kaufmann: San Francisco, CA, USA, 1994; pp. 121–129. [CrossRef]
12. Neumann, J.; Schnörr, C.; Steidl, G. Combined SVM-Based Feature Selection and Classification. *Mach. Learn.* **2005**, *61*, 129–150. [CrossRef]
13. Cho, J.; Hwang, I.; Oh, S. Vibration-Based Surface Recognition for Smartphones. In Proceedings of the 2012 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Seoul, Republic of Korea, 19–22 August 2012; pp. 459–464. [CrossRef]
14. Liu, J.; Chen, Y.; Gruteser, M. VibKeyboard: Virtual Keyboard Leveraging Physical Vibration: Demo. In Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking, New York, NY, USA, 3–7 October 2016; MobiCom'16; pp. 507–508. [CrossRef]
15. Jordan, M.I.; Mitchell, T.M. Machine learning: Trends, perspectives, and prospects. *Science* **2015**, *349*, 255–260. [CrossRef] [PubMed]
16. Pan, S.; Ramirez, C.G.; Mirshekari, M.; Fagert, J.; Chung, A.J.; Hu, C.C.; Shen, J.P.; Noh, H.Y.; Zhang, P. SurfaceVibe: Vibration-Based Tap & Swipe Tracking on Ubiquitous Surfaces. In Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks, New York, NY, USA, 18–21 April 2017; IPSN'17'; pp. 197–208. [CrossRef]
17. Lukowicz, P.; Ward, J.A.; Junker, H.; Stäger, M.; Tröster, G.; Atrash, A.; Starner, T. Recognizing Workshop Activity Using Body Worn Microphones and Accelerometers. In *Pervasive Computing*; Ferscha, A., Mattern, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 18–32. [CrossRef]
18. Deyle, T.; Palinko, S.; Poole, E.S.; Starner, T. Hambone: A Bio-Acoustic Gesture Interface. In Proceedings of the 2007 11th IEEE International Symposium on Wearable Computers, Boston, MA, USA, 11–13 October 2007; pp. 3–10. [CrossRef]
19. Stadler, S.; Riegler, S.; Hinterkörner, S. Bzzzt: When Mobile Phones Feel at Home. In Proceedings of the CHI'12 Extended Abstracts on Human Factors in Computing Systems, New York, NY, USA, 5–10 May 2012; CHI EA'12; pp. 1297–1302. [CrossRef]
20. Porzi, L.; Messelodi, S.; Modena, C.M.; Ricci, E. A Smart Watch-Based Gesture Recognition System for Assisting People with Visual Impairments. In Proceedings of the 3rd ACM International Workshop on Interactive Multimedia on Mobile & Portable Devices, New York, NY, USA, 22 October 2013; IMMPD'13; pp. 19–24. [CrossRef]
21. Poel, M. Detecting Mislabeled Data Using Supervised Machine Learning Techniques. In *Augmented Cognition. Neurocognition and Machine Learning*; Schmorrow, D.D., Fidopiastis, C.M., Eds.; Springer: Berlin/Heidelberg, Germany, 2017; pp. 571–581. [CrossRef]
22. Spolaôr, N.; Monard, M.C.; Tsoumakas, G.; Lee, H.D. A systematic review of multi-label feature selection and a new method based on label construction. *Neurocomputing* **2016**, *180*, 3–15. [CrossRef]
23. Liu, J.; Wang, C.; Chen, Y.; Saxena, N. VibWrite: Towards Finger-Input Authentication on Ubiquitous Surfaces via Physical Vibration. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, New York, NY, USA, 30 October–3 November 2017; CCS'17; pp. 73–87. [CrossRef]
24. Kefer, K.; Holzmann, C.; Findling, R.D. Evaluating the Placement of Arm-Worn Devices for Recognizing Variations of Dynamic Hand Gestures. *J. Mob. Multimed.* **2017**, *12*, 225–242.
25. Chen, W.; Lian, Y.; Wang, L.; Ruby, R.; Hu, W.; Wu, K. Virtual Keyboard for Wearable Wristbands. In Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems, New York, NY, USA, 6–8 November 2017; SenSys'17; pp. 44:1–44:2. [CrossRef]
26. Kim, Y.; Lee, W.S.; Raghunathan, V.; Jha, N.K.; Raghunathan, A. Vibration-Based Secure Side Channel for Medical Devices. In Proceedings of the 52nd Annual Design Automation Conference, New York, NY, USA, 7–11 June 2015; DAC'15; pp. 32:1–32:6. [CrossRef]
27. Ikeda, A.; Kosugi, S.; Tanaka, Y. Propagating Vibration Analysis of Leg towards Ankle Joint Angle Estimation. In Proceedings of the 24th International Conference on Intelligent User Interfaces: Companion, New York, NY, USA, 16–20 March 2019; IUI'19; pp. 11–12. [CrossRef]
28. McGrath, W.; Li, Y. Detecting Tapping Motion on the Side of Mobile Devices by Probabilistically Combining Hand Postures. In Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology, New York, NY, USA, 5–8 October 2014; UIST'14; pp. 215–219. [CrossRef]
29. Yurttadur, A.A.; Karaçay, T. Intrusion Detection Using Seismic Signals and Classification with Support Vector Machine. In Proceedings of the 5th International Conference on Mechatronics and Control Engineering, New York, NY, USA, 14–17 December 2016; ICMCE'16; pp. 103–107. [CrossRef]
30. Shen, D.; Markwood, I.; Shen, D.; Liu, Y. Virtual Safe: Unauthorized Walking Behavior Detection for Mobile Devices. *IEEE Trans. Mob. Comput.* **2019**, *18*, 688–701. [CrossRef]
31. Liu, J.; Chen, Y.; Gruteser, M. Sensing on Ubiquitous Surfaces via Vibration Signals: Poster. In Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking, New York, NY, USA, 3–7 October 2016; MobiCom'16; pp. 424–425. [CrossRef]
32. Miluzzo, E.; Varshavsky, A.; Balakrishnan, S.; Choudhury, R.R. Tapprints: Your Finger Taps Have Fingerprints. In Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services, New York, NY, USA, 25–29 June 2012; MobiSys'12; pp. 323–336. [CrossRef]

33. Liu, X.; Zhou, Z.; Diao, W.; Li, Z.; Zhang, K. When Good Becomes Evil: Keystroke Inference with Smartwatch. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, New York, NY, USA, 12–16 October 2015; CCS'15; pp. 1273–1285. [CrossRef]
34. Owusu, E.; Han, J.; Das, S.; Perrig, A.; Zhang, J. ACCessory: Password Inference Using Accelerometers on Smartphones. In Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications, New York, NY, USA, 28–29 February 2012; HotMobile'12; pp. 9:1–9:6. [CrossRef]
35. DeVaul, R.W.; Dunn, S. *Real-Time Motion Classification for Wearable Computing Applications*; Technical Report; MIT Media Laboratory: Cambridge, MA, USA, 2001.
36. Bayat, A.; Pomplun, M.; Tran, D.A. A Study on Human Activity Recognition Using Accelerometer Data from Smartphones. *Procedia Comput. Sci.* **2014**, *34*, 450–457. [CrossRef]
37. Srivastava, N.; Bhatnagar, M.; Yadav, A.; Dutta, M.K.; Travieso, C.M. Machine Learning Based Improved Automatic Diagnosis of Cardiac Disorder. In Proceedings of the 2nd International Conference on Applications of Intelligent Systems, New York, NY, USA, 7–9 January 2019; APPIS'19; pp. 11:1–11:8. [CrossRef]
38. Sherin, B.M.; Supriya, M.H. Selection and parameter optimization of SVM kernel function for underwater target classification. In Proceedings of the 2015 IEEE Underwater Technology (UT), Chennai, India, 23–25 February 2015; pp. 1–5. [CrossRef]
39. Saeys, Y.; Inza, I.; Larrañaga, P. A review of feature selection techniques in bioinformatics. *Bioinformatics* **2007**, *23*, 2507–2517. [CrossRef]
40. Arbabshirani, M.R.; Plis, S.; Sui, J.; Calhoun, V.D. Single subject prediction of brain disorders in neuroimaging: Promises and pitfalls. *NeuroImage* **2017**, *145*, 137–165. [CrossRef]
41. Mehrnezhad, M.; Toreini, E.; Shahandashti, S.F.; Hao, F. Stealing PINs via mobile sensors: Actual risk versus user perception. *Int. J. Inf. Secur.* **2018**, *17*, 291–313. [CrossRef] [PubMed]
42. Liu, J.; Wang, C.; Chen, Y. PIN Number-Based Authentication Leveraging Physical Vibration: Poster. In Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking, New York, NY, USA, 3–7 October 2016; MobiCom'16; pp. 426–427. [CrossRef]
43. Tran, D.S.; Yang, H.J.; Kim, S.H.; Lee, G.S.; Do, L.N.; Ho, N.H.; Nguyen, V.Q. Audio-Based Emotion Recognition Using GMM Supervector an SVM Linear Kernel. In Proceedings of the 2nd International Conference on Machine Learning and Soft Computing, New York, NY, USA, 2–4 February 2018; ICMLSC'18; pp. 169–173. [CrossRef]
44. Funato, N.; Takemura, K. Grip Force Estimation by Emitting Vibration. In Proceedings of the Adjunct Publication of the 30th Annual ACM Symposium on User Interface Software and Technology, New York, NY, USA, 22–25 October 2017; UIST'17; pp. 141–142. [CrossRef]
45. Önem, I.M. Testing and Improving the Performance of SVM Classifier in Intrusion Detection Scenario. In *Knowledge Discovery, Knowledge Engineering and Knowledge Management*; Fred, A., Dietz, J.L.G., Liu, K., Filipe, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 173–184. [CrossRef]
46. Bennett, K.P.; Campbell, C. Support Vector Machines: Hype or Hallelujah? *ACM SIGKDD Explor. Newsl.* **2000**, *2*, 1–13. [CrossRef]
47. Sangeetha, R.; Kalpana, B. Optimizing the Kernel Selection for Support Vector Machines Using Performance Measures. In Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India, New York, NY, USA, 16–17 September 2010; A2CWiC'10; pp. 1:1–1:7. [CrossRef]
48. Yao, J.; Zhao, S.; Fan, L. An Enhanced Support Vector Machine Model for Intrusion Detection. In *Rough Sets and Knowledge Technology*; Wang, G.Y., Peters, J.F., Skowron, A., Yao, Y., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 538–543. [CrossRef]
49. Hsu, C.W.; Chang, C.C.; Lin, C.J. *A Practical Guide to Support Vector Classification*; National Taiwan University: Taipei, Taiwan, 2016. Available online: http://www.csie.ntu.edu.tw/~cjlin (accessed on 9 November 2020).
50. Qin, Z.; Du, J.; Han, G.; Yong, G.; Guo, L.; Wang, L. LOL: Localization-free online keystroke tracking using acoustic signals. *Soft Comput.* **2019**, *23*, 11063–11075. [CrossRef]
51. Harrison, C.; Ramamurthy, S.; Hudson, S.E. On-Body Interaction: Armed and Dangerous. In Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction, New York, NY, USA, 19–22 February 2012; TEI'12; pp. 69–76. [CrossRef]
52. Strohmeier, P.; Carrascal, J.P.; Hornbæk, K. What Can Doodles on the Arm Teach Us about On-Body Interaction? In Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems, New York, NY, USA, 7–12 May 2016; CHI EA'16; pp. 2726–2735. [CrossRef]
53. Harrison, C.; Tan, D.; Morris, D. Skinput: Appropriating the Body as an Input Surface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, New Orleans, LA, USA, 30 April–5 May 2010*; Association for Computing Machinery: New York, NY, USA, 2010; pp. 453–462. [CrossRef]
54. Harrison, C.; Hudson, S.E. Scratch Input: Creating Large, Inexpensive, Unpowered and Mobile Finger Input Surfaces. In Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology, New York, NY, USA, 19–22 October 2008; UIST'08; pp. 205–208. [CrossRef]
55. Sato, M.; Poupyrev, I.; Harrison, C. Touché: Enhancing Touch Interaction on Humans, Screens, Liquids, and Everyday Objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, New Orleans, LA, USA, 5–10 May 2012*; Association for Computing Machinery: New York, NY, USA, 2012; pp. 483–492. [CrossRef]

56. Shi, Y.; Zhang, H.; Zhao, K.; Cao, J.; Sun, M.; Nanayakkara, S. Ready, Steady, Touch! Sensing Physical Contact with a Finger-Mounted IMU. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **2020**, *4*, 1–25. [CrossRef]
57. Edson-Niu. Butterworth-Filter. GitHub. 2018. Available online: https://github.com/nxsEdson/Butterworth-Filter (accessed on 31 January 2020).
58. Schölkopf, B.; Williamson, R.; Smola, A.; Shawe-Taylor, J.; Platt, J. Support Vector Method for Novelty Detection. In Proceedings of the 12th International Conference on Neural Information Processing Systems, Cambridge, MA, USA, 29 November–4 December 1999; NIPS'99; pp. 582–588.
59. Amer, M.; Goldstein, M.; Abdennadher, S. Enhancing One-Class Support Vector Machines for Unsupervised Anomaly Detection. In Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description, New York, NY, USA, 11 August 2013; ODD'13; pp. 8–15. [CrossRef]
60. Manevitz, L.M.; Yousef, M. One-Class Svms for Document Classification. *J. Mach. Learn. Res.* **2002**, *2*, 139–154.
61. Manevitz, L.M.; Yousef, M. Document Classification on Neural Networks Using Only Positive Examples (Poster Session). In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA, 24–28 July 2000; SIGIR'00; pp. 304–306. [CrossRef]
62. Devos, O.; Ruckebusch, C.; Durand, A.; Duponchel, L.; Huvenne, J.P. Support vector machines (SVM) in near infrared (NIR) spectroscopy: Focus on parameters optimization and model interpretation. *Chemom. Intell. Lab. Syst.* **2009**, *96*, 27–33. [CrossRef]
63. Vabalas, A.; Gowen, E.; Poliakoff, E.; Cassion, A.J. Machine learning algorithm validation with a limited sample size. *PLoS ONE* **2019**, *14*, e0224365. [CrossRef]
64. Prune. Selection Parameters in Libsvm. Web Page. 2016. Available online: https://stackoverflow.com/questions/37447619/selection-parameters-in-libsvm (accessed on 15 February 2021).
65. Gaspar, P.; Carbonell, J.; Oliveira, J.L. On the parameter optimization of Support Vector Machines for binary classification. *J. Integr. Bioinform.* **2012**, *9*, 33–43. [CrossRef]
66. Raff, E. What Is 'Eps' in Libsvm? Web Page. 2016. Available online: https://stackoverflow.com/questions/38996696/what-is-eps-in-libsvm (accessed on 15 February 2021).
67. Wang, L.; Xu, G.; Wang, J.; Yang, S.; Guo, L.; Yan, W. GA-SVM based feature selection and parameters optimization for BCI research. In Proceedings of the 2011 Seventh International Conference on Natural Computation, Shanghai, China, 26–28 July 2011; Volume 1, pp. 580–583. [CrossRef]
68. Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P. *Numerical Recipes in C; The Art of Scientific Computing*, 2nd ed.; Cambridge University Press: New York, NY, USA, 1992.
69. Tsai, S.F. Generating optimal order-of-addition designs with flexible run sizes. *J. Stat. Plan. Inference* **2022**, *218*, 147–163. [CrossRef]
70. McClelland, G.H. Optimal design in psychological research. *Psychol. Methods* **1997**, *2*, 3–19. [CrossRef]
71. Hoskins, D.; Turban, R.C.; Colbourn, C.J. Experimental Designs in Software Engineering: D-Optimal Designs and Covering Arrays. In Proceedings of the 2004 ACM Workshop on Interdisciplinary Software Engineering Research, New York, NY, USA, 5 November 2004; WISER'04; pp. 55–66. [CrossRef]
72. Hermann, M.; Christensen, H.; Reubsaet, J. Evaluation of essential parameters in the chromatographic determination of cyclosporin A and metabolites using a D-optimal design. *J. Pharm. Biomed. Anal.* **2002**, *30*, 1263–1276. [CrossRef] [PubMed]
73. Green, B.; Duffull, S.B. Prospective Evaluation of a D-Optimal Designed Population Pharmacokinetic Study. *J. Pharmacokinet. Pharmacodyn.* **2003**, *30*, 145–161. :1024467714170 [CrossRef] [PubMed]
74. Rajmohan, T.; Palanikumar, K. Modeling and analysis of performances in drilling hybrid metal matrix composites using D-optimal design. *Int. J. Adv. Manuf. Technol.* **2013**, *64*, 1249–1261. [CrossRef]
75. Vinayagamoorthy, R.; Rajeswari, N.; Vijayshankar, S.; Balasubramanian, K. Drilling Performance Investigations on Hybrid Composites by Using D-Optimal Design. *Int. Rev. Mech. Eng.* **2014**, *8*, 952–961. [CrossRef]
76. Kjeldskov, J.; Stage, J. New techniques for usability evaluation of mobile systems. *Int. J. Hum.-Comput. Stud.* **2004**, *60*, 599–620. [CrossRef]
77. Baillie, L.; Schatz, R. Exploring Multimodality in the Laboratory and the Field. In Proceedings of the 7th International Conference on Multimodal Interfaces, New York, NY, USA, 4–6 October 2005; ICMI'05; pp. 100–107. [CrossRef]
78. Carlsson, V.; Schiele, B. Towards Systematic Research of Multimodal Interfaces for Non-Desktop Work Scenarios. In Proceedings of the CHI'07 Extended Abstracts on Human Factors in Computing Systems, New York, NY, USA, 28 April–3 May 2007; CHI EA'07, pp. 1715–1720. [CrossRef]
79. De Sá, M.; Carriço, L. Defining Scenarios for Mobile Design and Evaluation. In Proceedings of the CHI'08 Extended Abstracts on Human Factors in Computing Systems, New York, NY, USA, 5–10 April 2008; CHI EA'08; pp. 2847–2852. [CrossRef]
80. Duh, H.B.L.; Tan, G.C.B.; Chen, V.H.h. Usability Evaluation for Mobile Device: A Comparison of Laboratory and Field Tests. In Proceedings of the Conference on Human-Computer Interaction with Mobile Devices and Services, New York, NY, USA, 12–15 September 2006; MobileHCI'06; pp. 181–186. [CrossRef]
81. Choi, S.; Hwang, J.; Cho, S.; Kim, A.; Nam, B.G. A low-power real-time hidden Markov model accelerator for gesture user interface on wearable devices. In Proceedings of the 2016 IEEE Asian Solid-State Circuits Conference (A-SSCC), Toyama, Japan, 7–9 November 2016; pp. 261–264. [CrossRef]
82. Foerster, F.; Smeja, M.; Fahrenberg, J. Detection of posture and motion by accelerometry: A validation study in ambulatory monitoring. *Comput. Hum. Behav.* **1999**, *15*, 571–583. [CrossRef]

83. Ravi, N.; Dandekar, N.; Mysore, P.; Littman, M.L. Activity Recognition from Accelerometer Data. In Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence, Pittsburgh, PA, USA, 9–13 July 2005; AAAI Press: Washington, DC, USA, 2005; IAAI'05; Volume 3, pp. 1541–1546.
84. ul Haq, M.E.; Awais Azam, M.; Asim, Y.; Amin, Y.; Naeem, U.; Khalid, A. Using Smartphone Accelerometer for Human Physical Activity and Context Recognition in-the-Wild. *Procedia Comput. Sci.* **2020**, *177*, 24–31. [CrossRef]
85. Qamar, N.; Siddiqui, N.; ul Haq, M.E.; Awais Azam, M.; Naeem, U. An Approach towards Position-Independent Human Activity Recognition Model based on Wearable Accelerometer Sensor. *Procedia Comput. Sci.* **2020**, *177*, 196–203. [CrossRef]
86. Kostakos, V.; Musolesi, M. Avoiding Pitfalls When Using Machine Learning in HCI Studies. *Interactions* **2017**, *24*, 34–37. [CrossRef]
87. Ruuska, S.; Hämäläinen, W.; Kajava, S.; Mughal, M.; Matilainen, P.; Mononen, J. Evaluation of the confusion matrix method in the validation of an automated system for measuring feeding behaviour of cattle. *Behav. Process.* **2018**, *148*, 56–62. [CrossRef] [PubMed]
88. Ben-David, A. Comparison of classification accuracy using Cohen's Weighted Kappa. *Expert Syst. Appl.* **2008**, *34*, 825–832. [CrossRef]
89. Kolesnyk, A.S.; Khairova, N.F. Justification for the Use of Cohen's Kappa Statistic in Experimental Studies of NLP and Text Mining. *Cybern. Syst. Anal.* **2022**, *58*, 280–288. [CrossRef]
90. Vieira, S.M.; Kaymak, U.; Sousa, J.M.C. Cohen's kappa coefficient as a performance measure for feature selection. In Proceedings of the International Conference on Fuzzy Systems, Barcelona, Spain, 18–23 July 2010; pp. 1–8. [CrossRef]
91. McHugh, M.L. Interrater reliability: The kappa statistic. *Biochem. Med.* **2012**, *22*, 276–282. [CrossRef]
92. Ribeiro, M.T.; Singh, S.; Guestrin, C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 13–17 August 2016; KDD'16; pp. 1135–1144. [CrossRef]
93. Linardatos, P.; Papastefanopoulos, V.; Kotsiantis, S. Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy* **2021**, *23*, 18. [CrossRef]
94. Lin, H.T.; Lin, C.J. *A Study on Sigmoid Kernels for SVM and the Training of Non-PSD Kernels by SMO-Type Methods*; Technical Report; Department of Computer Science and Information Engineering, National Taiwan University: Taipei, China, 2003.
95. Yang, X.S. A New Metaheuristic Bat-Inspired Algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*; González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 65–74. [CrossRef]
96. Mohammed, L.B.; Raahemifar, K. Improving Support Vector Machine Classification Accuracy Based on Kernel Parameters Optimization. In Proceedings of the Communications and Networking Symposium, San Diego, CA, USA, 15–18 April 2018; CNS'18; pp. 1–9.
97. user3421031. Android Accelerometer Sampling Rate/Delay Stabilization. Web Page. 2014. Available online: https://stackoverflow.com/questions/23302645/android-accelerometer-sampling-rate-delay-stabilization (accessed on 18 November 2020).
98. Guerra-Casanova, J.; Sánchez-Ávila, C.; Bailador, G.; de Santos Sierra, A. Authentication in mobile devices through hand gesture recognition. *Int. J. Inf. Secur.* **2012**, *11*, 65–83. [CrossRef]
99. Choi, H.R.; Kim, T.Y. Directional Dynamic Time Warping for Gesture Recognition. In Proceedings of the 2017 2nd International Conference on Multimedia Systems and Signal Processing, New York, NY, USA, 13–16 August 2017; ICMSSP 2017; pp. 22–25. [CrossRef]
100. Yuan, J.; Lin, Q.; Zhang, W.; Wang, Z. Locally Slope-Based Dynamic Time Warping for Time Series Classification. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, New York, NY, USA, 3–7 November 2019; CIKM'19; pp. 1713–1722. [CrossRef]
101. Kuhfeld, W.F.; Tobias, R.D.; Garratt, M. Efficient Experimental Design with Marketing Research Applications. *J. Mark. Res.* **1994**, *31*, 545–557. [CrossRef]
102. Kuhfeld, W.F.; Tobias, R.D.; Garratt, M. Efficient Experimental Design with Marketing Research Applications (Revision). PDF. 2010. Available online: https://support.sas.com/techsup/technote/mr2010d.pdf (accessed on 6 November 2020).
103. StatguyUser. How to Interpret D-Efficiency in Experimental Design in R optFederov. Web Page. 2015. Available online: https://stats.stackexchange.com/questions/137695/how-to-interpret-d-efficiency-in-experimental-design-in-r-optfederov (accessed on 25 March 2020).