Toward Designing Usable Body-Based Gesture Interactions

A Dissertation

Presented to

the Faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Doctor of Philosophy (Computer Science)

by

Md Aashikur Rahman Azim

April 2024

Abstract

The integration of wearables into our lives has revolutionized the way we interact with technology. A diverse array of form factors, including glasses, earphones, rings, watches, pendants, and VR headsets, are now equipped with advanced processors and sensors, facilitating effortless communication with other smart devices. This integration has given rise to quick microinteractions, with users relying on body-based gestures to perform various tasks. However, as we increasingly rely on body-based gestures to interact with technology, ensuring their usability and security becomes crucial.

Designing and implementing usable body-based gestures involves a multifaceted challenge that requires a fine balance between minimizing false activations and ensuring the gestures are intuitive and user-friendly. This balance is crucial for creating gestures that are not only easy to perform and remember but also socially acceptable and independent of specific devices. A good gesture is characterized by its ease of execution, memorability, compatibility across various devices, accurate recognizability with a minimal error rate, and social acceptability. Additionally, maintaining the integrity of the performed gesture is paramount, highlighting the importance of the gesture's reliability and consistent interpretation by the system.

This dissertation addresses the challenges in the design and implementation of user-friendly, memorable, and false-activation-resistant body-based gestures. It introduces *SequenceSense*, a tool that empowers gesture designers to easily modify gestures, assess recognition performance, and pinpoint potential false activations without resorting to extensive data gathering or experimental efforts. Addressing gesture compatibility and reusability across devices, the dissertation presents *UnifiedSense*. This novel method facilitates the detection of device-dependent gestures using sensors from various wearable devices, even in the absence of the device originally intended for gesture detection. Finally, to tackle the gesture integrity of the performed gesture, this dissertation proposes *ManipulaSense*, an *Autoencoder*-based anomaly detection technique that leverages users' inherent hand movements to identify manipulations of the hand movements, thereby preserving the integrity of application use.

This research significantly improves usability, efficiency, and reliability in gesture interactions for wearable devices, promoting their broader adoption and enhancing the overall user experience.

Approval Sheet

This dissertation is submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

Md Aashikur Rahman Azim

This dissertation has been read and approved by the Examining Committee:

Felix Xiaozhu Lin, Committee Chair

Seongkook Heo, Advisor

Sara Lu Riggs

Brad Campbell

Tariq Iqbal

Accepted for the School of Engineering and Applied Science:

Jennifer L. West, Dean, School of Engineering and Applied Science

April 2024

Acknowledgements

I am grateful to many people who have contributed to shaping this dissertation. This dissertation would not have been possible without the influence, advice, and support of many colleagues, friends, and family.

First and foremost, I express my profound gratitude to Allah, the creator of everything, for enriching me with the strength to overcome challenges and steadfastly pursue my journey to its completion. I am immensely thankful for the wisdom imparted upon me and for the health and well-being sustained throughout my Ph.D. endeavors.

Secondly, I want to express my gratitude to my advisor, Prof. Seongkook Heo, for his unwavering support, patience, and mentorship throughout my Ph.D. life. He inspired me to set ambitious goals and thoroughly develop my papers, offering me the freedom to explore and learn at my own pace. Despite the bunch of initial rejections, his approach allowed me to publish without the pressure of immediate results, focusing on quality work over publication count. His guidance also opened doors to valuable collaborations, enhancing my skills and broadening my perspectives in the research field.

Thirdly, I would like to express my gratitude to my Ph.D. committee members, Prof. Felix Lin, Prof. Brad Campbell, Prof. Sara Riggs, and Prof. Tariq Iqbal, for their insightful feedback on my Ph.D. proposal. Their detailed advice and critiques significantly contributed to refining and improving my dissertation, making it more coherent and impactful.

Fourthly, I would like to extend my heartfelt gratitude towards my co-authors, including Erzhen Hu, Adil Rahman, Wen Ying, Zihao Su, and Peiyu Zhang; without their effort, dedication, and input, none of my work would be possible. They taught me valuable lessons in various technical subjects. I will forever miss the meetings and conversations we had while we were solving difficult problems in both academia and personal life. Most importantly, they are a great source of inspiration and encouragement that help me reach the finish line.

Finally, heartfelt thanks go to my family. My mother, parents-in-law, and wife were the ones who encouraged me to begin this Ph.D. journey. Despite moments of doubt and thoughts of giving up, they supported me during those hard times, reminding me of the value of perseverance. Today, I recognize that

earning my Ph.D. has been a life-changing journey, enriching me with invaluable lessons for the future. My son, Numair, deserves special mention for being my steadfast cheerleader, enduring my frustrations gracefully, and celebrating every step of progress with me. In the end, I would like to dedicate my Ph.D. to my late father and educator, Md Haider Ali, who always encouraged me to read, learn, strive for societal betterment until 2009. His influence remains a foundational aspect of my academic journey.

Pursuing a Ph.D. degree is difficult. Looking back, it feels like an endless sequence of hurdles, yet sprinkled with rewarding moments. I'm truly pleased to have completed what I set out to do six years ago and am excited about the upcoming opportunities in both my personal and professional life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and Scope

The rise of wearable technology and gesture-based interactions indicates a significant transformation in human-computer interaction (HCI), reshaping our engagement with digital systems through intuitive and natural body movements. Wearables such as glasses, earphones, rings, watches, pendants, and VR headsets are embedded with advanced processors and sensors, facilitating direct communication with an array of smart devices. This technological evolution empowers users to employ diverse body-based gestures for interacting with technology in a manner that is both instinctive and seamless [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19], promoting uses across various platforms for enhanced convenience, efficiency, and social acceptability. For example, Spotify allows users to control the music playing on smart speakers connected to a smartphone by using controls on their smartwatches [20], and researchers proposed using a wearable device as an input device for another wearable device [21, 22]. Ensuring the usability of body-based gestures is essential for maintaining user satisfaction and the widespread adoption of wearable technologies. Several challenges arise when designing and implementing usable gesture interactions. This thesis presents an overview of the challenges involved and outlines the research questions addressed for designing usable gestures.

I start with defining a **good gesture** aligned with the state-of-the-art gesture design. A good gesture, within the context of designing usable body-based gestures, is defined by several critical properties. It must be comfortable to perform, ensuring ease of execution [23]; memorable, facilitating easy recall by users [1, 24]; compatible across different devices, allowing for a consistent user experience irrespective of the specific technology in use [23]; accurately recognizable, maintaining a balance between false positive[1] and false

[1]False positives occur when a user does not intend to perform a gesture, but the system recognizes one anyway.

negative[2] errors [23]; and socially acceptable, not drawing undue attention in public settings [1, 25]. Moreover, the integrity of the performed gesture is paramount, underscoring the importance of reliable and consistent interpretation by the system. Figure 1.1 illustrates the properties of a good gesture. Here, I introduced a new gesture property (i.e., *gesture integrity*) in this thesis.

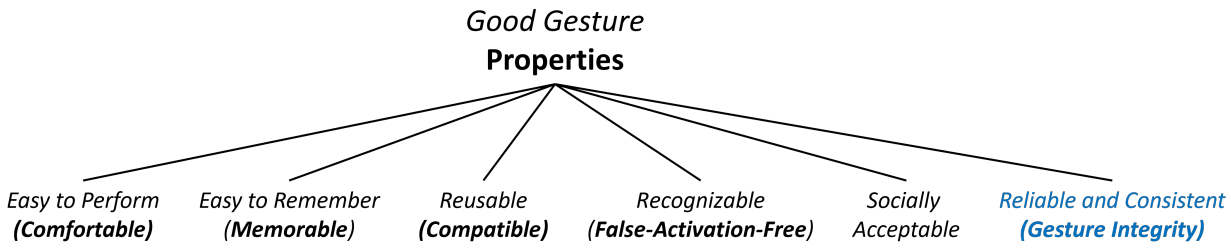


Figure 1.1: Properties of a *Good Gesture*

Designing body-based gestures for user interfaces presents a multifaceted challenge that revolves around achieving a balance between reducing false activations and ensuring the gestures are simple and user-friendly. This requires a repetitive approach to balancing false positive errors against the need for gestures to be memorable, socially acceptable, and easily executable. Secondly, the same gesture should be compatible across devices to enhance its reusability. Moreover, a good gesture design should consider the consistency and reliability of the gestures, which is, a performed gesture should result in a consistent outcome all the time it is being performed by a user. Next, I discuss each of the challenges in more detail.

The first challenge revolves around a balance between reducing false activations and ensuring simplicity and user-friendliness [26]. On one hand, complex gesture designs may help mitigate false activations, but they introduce additional hurdles related to memorability [27, 24] and social acceptability [28, 1]. Complex gestures are often difficult to perform and remember, diminishing their overall usability. Moreover, such gestures may attract unwanted attention from others, rendering them socially unacceptable in public settings. In contrast, the goal is to implement simple and user-friendly gestures that can be easily executed and remembered by regular users. However, achieving simplicity and user-friendliness while minimizing false activations presents a significant challenge [29]. Designers must thoughtfully consider these factors to create intuitive and efficient gesture interactions that seamlessly integrate with users' everyday activities. The process entails careful adjustments to the gesture recognizer and modifications to gestures, which can involve extensive manual tweaking and rigorous validation studies in both controlled laboratory settings and real-world contexts [23, 30, 31]. Effectively addressing this challenge allows for an optimal balance between gesture recognition accuracy, usability, and user satisfaction, facilitating the efficient iteration and advancement of gesture design.

[2]False negatives occur when a user has performed a gesture correctly, but the system does not recognize it.

Secondly, the principles of compatibility are paramount in gesture design, as they facilitate user comprehension of action effects, thereby mitigating frustration and elevating the interaction experience [32]. To this extent, once users learn a certain gesture to perform a specific task, the gesture interactions should facilitate users with the same outcome when they perform the same gesture. The designers have to ensure this compatibility to help users reuse their muscle memory, which is what users learn through repetitively performing a gesture for specific tasks. For example, if a user used to perform swipe-right gestures on their earphones to receive a call on their phones, this gesture should facilitate the user all the time, even if the user wears different earphones or is not even wearing earphones. However, these compatibility are not present in the current gesture interactions since many gesture recognition systems are closely tied to specific devices or sensors. This dependency restricts users to perform gestures only when wearing or utilizing those specific devices, hindering the efficiency and flexibility of interaction. Users may experience frustration if they attempt to perform a gesture while wearing non-smart devices or if they forget to wear their smart devices. To enhance usability, it is necessary to explore methods that allow gestures to be independent of the specific devices used, enabling users to interact seamlessly across a range of wearable devices.

Mitigating false activations and ensuring device-independent design are crucial for facilitating usable gesture interactions when gestures are detected in a discrete manner. However, the usability of gestures while the system needs to detect the gestural interaction continuously might add a burden to the designer to make it usable and secure for the users. For example, in-air hand interactions are prevalent in Virtual Reality (VR), and people are using hand gestures for continuous interactions with different applications. Prior studies have shown that manipulating the visual movement of the hand to be different from the actual hand movement, i.e., perception manipulation (PM), could create a more immersive and engaging VR experience. However, this manipulation risks degrading task performance and, if maliciously applied, poses a threat to user safety [33, 34, 35]. For example, it might impact users' gained muscle memory of what they learned through repetitive usage of gestures. Since manipulations might negatively affect users while performing in-air hand gesture interactions, it is important for the designers to ensure the usability of these performed gestures in terms of making the environment secure. Understanding the implications of PM on gesture interactions in VR is also crucial for creating secure and user-friendly experiences. Hence, this thesis delves into the investigation of this challenge and its effects on user performance.

There are many dimensions of designing usable body-based gestures, and this dissertation explores the following aspects of them:

1. How a framework be designed and developed to help designers efficiently design gestures that are free of false activations with minimal effort, ultimately benefiting the users;

2. How a system can be developed to allow users to seamlessly reuse their gestures across different devices by utilizing the muscle memory they have developed;

3. How perception manipulation affects task performance while performing gestures, particularly in virtual environments;

4. How a system be developed to ensure that gestures performed by users produce consistent and reliable outcomes.

1.2 Contributions

I have developed a set of solutions addressing the challenges mentioned earlier. The primary issue addressed is the efficient design of body-based gestures, enabling designers to create user-friendly, false-activation-free gestures. By providing tools that facilitate efficient gesture creation, I directly benefit users by ensuring the designers can produce usable gestures, thereby enhancing the system's adaptability for the users. Designers often resist engaging in the intricate process of gesture design due to its time-consuming and labor-intensive nature. This reluctance can lead to the development of impractical gestures, resulting in user frustration, decreased system usage, and, eventually, abandonment of the application.

Existing methods like Mogeste [36] and MAGIC [1] have been developed to facilitate the design of usable body- and motion-based gestures. However, these methods have limitations. Mogeste allows designers to create and test gestures in real-world scenarios but still requires testing in realistic scenarios for usability evaluation and re-collection of samples for recognizer training when modifying gestures. MAGIC provides insights into gesture performance and potential conflicts but requires the recollection of samples and modifications for usability assessment. To address these challenges, *SequenceSense* is developed as an efficient tool for constructing robust body-based gesture sets. It utilizes a sequence-based gesture recognizer and an automatic conflict analyzer, allowing designers to easily modify gestures and evaluate recognition performance without re-collecting samples or conducting experiments. SequenceSense prioritizes gesture usability over robustness. I validated SequenceSense's efficacy in designing robust gestures with low false positives through a user study with nine gesture designers.

Secondly, to achieve device-independent gesture interactions, I developed *UnifiedSense*. UnifiedSense is a system that enables device-independent gesture interactions by leveraging sensors on wearable devices. It trains a recognition model using gestures detected on a reliable primary device and collects sensor data from all available devices. I named this process as "*over-the-shoulder training*" method. This approach eliminates the need for explicit training for each device. The over-the-shoulder training method automatically trains

the model by collecting samples as users perform gestures on their wearable devices. Users can use gestures associated with a primary device even when they are not wearing it. While researchers have previously explored the concept of indirectly sensing gestures using user-worn devices, such as using a mobile device in the pocket to detect foot gestures [10] or earbuds to detect touch gestures on the face [37], the diversity of wearable devices and user configurations makes it impractical for users to train the model for each new device or combination of devices. In contrast, the proposed over-the-shoulder training method enables the system to automatically train the model by collecting training samples as users perform gestures on their wearable devices. I conducted a technical evaluation with data collected from 15 participants with four types of wearable devices. It showed that UnifiedSense could correctly recognize 5 gestures (5 gestures $\times$ 5 configurations) with an accuracy of 90.9% ($SD = 1.9\%$) without the primary device present.

In addition to addressing the challenges of designing usable body-based gestures, this research investigated the effect of perception manipulation on the learning of optimal reaching movement in VR through two user studies where participants completed simple tasks with varying levels of orientation and magnitude manipulation on the user's hand movement. Our findings indicate that, while the task completion times were not different by the existence of the manipulation, the movement behavior showed differences. Participants used more corrective movements to complete the tasks, as the accuracy of the initial ballistic movement was worse when the level of manipulation was higher.

Finally, to enhance the usability of the performed gestures through real-time feedback, I developed *ManipulaSense*, an *Autoencoder*-based anomaly detection technique that leverages users' inherent hand movements to identify perception manipulation, thereby preserving the integrity of application use. Our model is trained on regular (i.e., non-manipulated) hand movement patterns and employs a stochastic thresholding approach for anomaly detection. I validated our method through a technical evaluation involving 21 participants engaged in reaching tasks under manipulated and non-manipulated scenarios. The results demonstrated a high accuracy of perception manipulation detection at 93.7%, with an F1-score of 93.9%.

In summary, this dissertation makes the following major contributions:

- Development of *SequenceSense*, a gesture designing tool that enables efficient modification and evaluation of gestures to minimize false activations, without extensive data collection or experiments. SequenceSense facilitates user-friendly, easy-to-perform, and memorable gesture design with minimal effort, ultimately benefiting the general users. Our system is made publicly available here.[3]

- Introduction of *UnifiedSense*, a system that enables device-independent gesture interactions by leveraging sensors on other wearable devices, ensuring seamless interaction even when the primary detecting device

[3] https://github.com/aashikazim/SequenceSenseUI

is unavailable. UnifiedSense facilitates the reusability of the gestures across devices and promotes the compatible gesture design. The dataset used to build UnifiedSense is publicly available here.[4]

- Investigation into the impact of perception manipulation on user experience and task performance during gesture interactions in VR environments, providing insights into the effects of manipulation on gesture execution. *This dataset will be shared with other researchers upon request.*

- Introduction of *ManipulaSense*, a novel deep-learning approach for detecting the presence of perception manipulation in VR from user's hand movement behavior. To our knowledge, this is the first method to automatically detect the perception manipulation in VR from the user's behavior data. ManipulaSense facilitates users in knowing the status of the gestures being performed, thereby enhancing their integrity. The dataset used to build ManipulaSense is publicly available here,[5] addressing the absence of publicly available datasets for perception manipulation research.

1.3 Thesis Statement

Leveraging heterogeneous sensor data from wearable smart devices enables the design and implementation of secure, compatible, and conflict-free body-based gesture interactions with minimal effort.

1.4 Dissertation Outline

The organization of the rest of this dissertation is given below.

- Chapter 2 first discusses different body-based gesture interactions. Then, it discusses the state-of-the-art works related to designing usable gestures and tools, the challenges of mitigating false activations, and their current state-of-the-art works. Next, I discuss wearable cross-device interactions and the state-of-the-art of wearable gesture recognition algorithms. Finally, I discuss the pros and cons of perception manipulation of the in-air hand movement in various contexts and discuss the state-of-the-art anomaly detection techniques in relation to our approach to perception manipulation detection.

- Chapter 3 describes *SequenceSense*, a tool developed to help designers efficiently build a usable foot-based gesture set using a sequence-based gesture recognizer and an automatic conflict analyzer.

- Chapter 4 proposes a new method, *UnifiedSense*, to enable device-dependent gestures even when the device that detects such gestures is missing by utilizing sensors on other wearable devices.

[4] https://github.com/aashikazim/UnifiedSense_DataSet
[5] https://github.com/aashikazim/perception-manipulation-dataset

- Chapter 5 presents the results of our investigation of perception manipulations while performing reaching tasks in VR through in-air gestures.
- Chapter 6 proposes *ManipulaSense*, an autoencoder-based anomaly detection technique that can be integrated with the current VR system to enhance the usability of using gesture by making the environment secure for the user through real-time feedback of the performed gestures.
- Chapter 7 summarizes the contributions presented in this dissertation. It concludes the dissertation by providing directions for future research.

Chapter 2

Related Work

2.1 Body-Based Gesture Interactions

Prior research has developed many techniques for body-based gesture inputs. Here, I discuss gestural interactions that use different parts of the body.

2.1.1 Using the Arm, Wrist, and Fingers

For arm-based inputs, significant research has explored sensing hand or finger postures through Electromyogram (EMG) [38, 13], camera vision [5, 39], and acoustic sensors [15, 16, 14]. Crossan et al. showed wrist rotation could be used for input in stationary situations [40], while Costanza et al. looked into isometric upper arm contractions for unobtrusive and socially acceptable interactions [41].

As the human wrist is dexterous with flexion-extension, radioulnar deviation, and rotation movements [40, 42], some approaches such as WristWhirl [43] and "With a Flick of the Wrist" [44] explore the idea of leveraging the wrist joint as a controller. Indeed, many of the distal pointing related tasks [45, 46] involve the combination of the arm and wrist motions. This also gives rise to text entry techniques based on wrist tilting [47, 48, 49] and tilt-based gesture typing [50, 51].

The growing availability of smartwatches and smart rings have prompted researchers to explore new forms of human-computer interaction and augmentation. Duet [52], VibRing [53] and Expressy [54] leverage the smartwatch or smart ring to add expressiveness to touch-based interactions on a phone or tablet device. Digital digits [55] and "Ring Form Factor" [56] surveyed and explored the design space of smart rings for interaction. Webb et al. [57] [64] explored using wearable devices such as a smart ring as context for surface and pen-based interaction. Magic Finger [58] and TouchCam [59] enables users to interact with surfaces

using optical sensor embedded in a wearable ring form factor. DeformWear [60] is a tiny finger worn device that enables precise input using pressure, shear and pinch deformations. PredicTouch [61] and Moschetti et al. [62] explore the combination of wrist and finger IMU for reducing touch screen latency and for recognition of daily gestures. SynchroWatch [63] uses rhythmic correlation between a user's thumb movement (magnet ring) and on-screen blinking controls (smartwatch).

2.1.2 Using the Head, Eyes, and Face

Interactions using the head, eyes, and face have also been studied. Head movements, such as orientation and nodding [64] or left and right tilting [11], have been used for hands-free input. The eyes are a common interaction modality for those with limited motor movements [65]. The nose has been used for hands-free indirect pointing [66] and to perform touch interactions [67, 68]. The physical movement of the tongue [69, 70] and blowing with one's mouth [71, 72] have also been effective for input.

For able-bodied users, head movement is a valuable interaction channel. Studies have shown that sensing head orientation and position can help the calibration of gaze interaction and promote the accuracy of gaze-based selection tasks [73, 74]. Head movements were also leveraged to control desktop cursors [75, 66] and mobile devices [11], by mapping the position of the head to the cursor. Head gestures were also proposed for performing discrete operations on the desktop [76, 77, 78] and HMD glasses [79, 80]. HeadTurn [77] enables users to adjust input numeric values by turning their heads left or right beyond the range threshold. HeadPager [78] enables users to turn pages in two directions by leaning their heads to the left or the right area. HeadNod [76] supports quick dialogue answering via a nod or shake of the head. Glassgesture [79] was the first work to leverage head gestures to achieve user authentication on AR headsets. Smoothmoves [80] requires users to follow the movement trajectory of the target with the head to select it, on AR headsets.

Besides the preceding techniques only leveraging head movements, previous research also studied to combine the use of gaze and head movements [81, 82, 83]. As gaze can reflect the focus and intention of the user [84], it has been naturally leveraged as an input method [85]. Additionally, gaze changes have a strong correlation with head movement [86], this can promote the recognition of head orientation and head gestures [87]. Studies [82, 83] also showed that by combining the use of gaze and head movement data, target selection techniques can achieve higher performance than using only one of them (faster than head pointing, more accurate than eye pointing).

2.1.3 Using the Feet

Many researchers have been exploring interaction techniques with the foot. Pearson et al. [88] proposed a foot-operated cursor-positioning device called the “planar slide mole" and assessed its performance against a mouse for target selection. Takeuchi [89] proposed foot tap gestures to operate a navigating map system. Scott et al. [10] evaluated the use of four foot gestures: lifting a toe/heel and foot rotations pivoting the heel/toe. They exploited the mobile's accelerometer placed in the pocket to determine the gestures. As an example of systems using footwear pressure sensors, Paradiso et al. [90] proposed a dance system where a user can control background music by using foot-based gestures. As another example of a pressure-sensor-based foot gesture, Papetti [91] proposed a sandal-placed foot-tapping interface with audio-tactile feedback. Yin and Pai [92] showed a user-controlled virtual character using only foot gestures (specifically foot pressure). However, the authors mention that it is impossible to recreate full-body motion by using foot pressure alone. Lv et al. [93] presented an augmented foot interaction interface that detects and tracks the user's foot motion and foot gestures using computer-vision-based algorithms. Two implementations were presented in the paper, an augmented football game and an augmented foot piano, both controlled by foot pressure or gestures only. Another study suggested controlling a mobile phone menu by “kicking" the wanted option [94]. Han et al. [95] explored how to control various mobile actions such as navigation and zoom with kick gestures. Moreover, few researchers have demonstrated the use of foot gestures as an interaction mechanism alongside other input modalities such as hand. For example, Johannes et al. [96] demonstrated how multi-touch hand gestures, in combination with foot gestures, can be used to perform navigation tasks in interactive systems. They showed an application of combining foot (indirect input) with hand (direct input) as a multimodal input for Geographic Information System (GIS). They applied foot pressure input to navigate (with pan, zoom, rotate, and tilt) a spatial document. Ricardo et al. [97] represented the feet interaction mechanism with vertical surfaces. They implemented interaction techniques for interacting with the bottom part of vertical displays, where the hands would not be able to reach typically. Fukahori et al. [30] presented a user-defined gesture study of foot pressure and pattern recognition. Recently, Daniel and Cooperstock [98] have shown that foot can compare with traditional hand-based interfaces in a pointing task based on Fitts's law task, by using variable friction to assist users in reducing pointing overshoot and increasing accuracy. Besides, several types of research have proposed foot-based interaction techniques to move around the virtual reality (VR) environment [99, 100, 90, 101]. To be more precise, NCSA's CyberBoots use a pressure sensor array in an overshoe to provide walking interaction in a virtual reality environment [99]. Haan et al. used foot gestures to present simplified locomotion [100]. ShoeSoleSense [90] presents a navigating system in virtual environments. Müller et al. [101] investigated direct and indirect foot-based interaction with virtual contents.

2.1.4 Using the Full Body

Full-body interaction includes the use of body movements and gestures for interacting with computers, which may be categorized into four kinds: (i) full-body only (e.g., Kinect), (ii) full-body plus external devices, (iii) external device only (e.g., Wii Remote), and (iv) body-centric interaction (i.e., using the body as interaction space). In full-body only interaction, users interact with computers (e.g., TV, games) using full-body movements and gestures through motion-camera sensing devices. To enhance realism, full-body only interaction may be augmented with external devices such as artificial gun [102] for interaction. In external device only interaction, body movement is recognized through controllers (e.g., Wii Remote) or mobile sensors'[103], rather than from a motion-camera sensing device. In body-centric interaction, designers explored extending the interaction space to the body. For example, Chen [104] investigated "body-centric interaction" for mobile phones which extends the mobile device interaction space from only the 2D screen to body space, e.g., using the arm as menus. Shoemaker et al. [105] also conducted similar work but on a large display, e.g,. touching the hip to open a toolbar on a map application. Harrison et al. [106] investigated how the body can become the input space, e.g., using the palm as a phone input screen.

A large body of work has been conducted regarding full-body interaction. Bianchi-Berthouze [107] investigated how body movement affects game engagement, and found that body movement enhances affective and social experiences in gameplay. In relation to movement, Isbister et al. [108] suggested that a greater amount of movement appears to lead to a greater amount of enjoyment. Nijhar et al. [109] found that increasing movement recognition precision leads to player's higher level of immersion. Pasch et al. [110] identified movement-related factors that influence player engagement: natural control, mimicry of movements, proprioceptive feedback, and physical challenge. Tholander and Johansson [111] studied what makes movement enjoyable by observing non-digital artefacts (e.g., real-world sports) and proposed eight design guidelines, e.g., allowing users to significantly improve their skills by allowing only small changes in movement could increase their sense of pride and mastery. Gerling et al. [112] explored full-body interaction for the elderly. Some suggested guidelines include fatigue management, accessibility support, simple setup and easy gesture recall. Isbister and DiMauro [113] analyzed movementbased design patterns such as kinesthetic mimicry and piecemeal movement. Overall, few studies have considered the simultaneous use of game gestures.

Regarding the application area of full-body interaction, it has greatly expanded in recent years, and includes using full-body interaction for interaction with large public displays [114, 115], TV controls [116], storytelling [117] and collaborative virtual environments [118]. A recent research effect focuses on using full-body interaction for promoting physical well-being (i.e., exergames). Some research findings suggest that exergames motivate users in physical activity [119, 120, 121] and enhance physical health [122, 123].

Exergames have also been used to address serious health problems such as cerebral palsy [124], stroke [125], and chronic back pain [126].

2.2 Designing Conflict-Free Body-Based Gestures

Over time, to assist gesture designers in creating effective gestures, numerous design tools have been developed by researchers. This section will explore various gesture design tools, highlighting their advantages and disadvantages. Additionally, it will cover strategies researchers have employed to reduce false activations in gesture design.

2.2.1 Gesture Design Tools

Gesture design is a time- and effort-requiring process. Researchers have been developing tools to make the process more efficient and more accessible for gesture designers without expertise in developing gesture recognizers. Early research on gesture interfaces focused on the design of tools to improve the specification, mapping, and recognition of gestures by providing gesture templates or allowing designers to demonstrate gestures [127, 128, 129, 130, 131, 132]. While these tools help designers easily design gestures and build gesture recognizers without programming, evaluating gesture performances, such as gesture classification accuracy and the number of false activations, requires conducting a user study.

To expedite the gesture design process by reducing the need for conducting user studies, some gesture design tools provide gesture performance prediction. For example, Hartmann et al. [133] developed Exemplar which assists the designers in the fast prototyping of gesture interactions by connecting real-world hardware components with the Exemplar. Once the hardware is connected, the designer trains the system by providing examples of samples to be recognized. However, the system does not differentiate between testing and actual use and therefore does not provide information on performance with respect to between-class differentiability. Such information is essential when designing multiple gestures that should not interfere with one another, and most importantly with daily activities. MAGIC proposed by Ashbrook and Starner [1] tried to reduce the gap between gesture interference within classes, and also with daily activities. Akin to *Exemplar*, MAGIC considers information about consistency between samples, the chances of getting false activations, and the distinctiveness of classes of gestures, all of which are used to improve the recognition rate of the motion gestures created by designers. In MAGIC 2.0 [134], Kohlsdorf et al. extended MAGIC to better account for the false positive motions that could occur in everyday life. By integrating indexable *Symbolic Aggregate* approximations, a designer could search through a database of everyday motions to determine if a given gesture could be accidentally triggered. Although MAGIC and MAGIC 2.0 facilitate false positive detection

in the process of gesture design, the designers have to reiterate throughout the process as these systems do not support modification of the already designed gesture for further false positive checking. Subsequently, Mogeste [36] developed by Parnami et al. is a mobile tool to allow gesture designers to design, modify, and test motion gestures anywhere. By letting gesture designers be at the site where the gestures will be used and test and modify the gestures at the site, Mogeste can help them identify potential gesture conflicts and accelerate the gesture design process. However, it still requires the designers to test the gestures in a realistic scenario to evaluate the usability, and modifying gestures needs recollection of gesture samples to train the recognizer.

Studies have shown that visualizing sensor data and using graphical markup language may allow easy and exploratory gesture prototyping for gesture designers [135, 136]. As the continuous sensor data is visualized as trajectories on display in real-time, these tools can help gesture designers to identify the patterns and try alternative gestures. The metaphors used in these systems, such as the hurdles [135] and the mass-spring model [136], also made the recognition algorithm easier for the designers to understand than machine-learning-based recognizers [135], although this limits the types of usable features. While this method allows the designers to quickly experiment with different gestures, this does not provide the gesture performance analysis, making a formal user study necessary to understand the performance.

The above-discussed tools were developed to make the motion gesture design process more efficient, particularly for gesture systems that use IMU data. Previous studies have also shown that the approaches that are useful for IMU-based gesture sensing systems could be useful for gesture systems that use other sensors, such as depth cameras that track user's body movement. For example, CUBOD [137] developed by Tang and Igarashi follows a similar approach to MAGIC; it shows the prediction of classification performance, false activations, and gesture consistency based on gesture *Goodness Measure* so that the designers can know whether the gesture is good or bad without conducting the study. GestureAnalyzer [138] proposed by Jang et al. provides interactive clustering and visualization techniques to help gesture designers categorize and characterize gestures collected during gesture elicitation studies. Similar to the previous IMU-based approaches that visualize the motion data [133, 1], the study found that the visualization of gesture features could help designers better understand the gesture characteristics, including the similarities of the designed gestures. However, modifying the gestures when potential conflicts were found still remains as a challenge.

To summarize, many attempted to make the gesture design process easier and more efficient by developing a user-friendly design tool [36, 1, 133], easier recognizer modification [136, 135], and a database of everyday motions for false positive testing [1, 134, 137]. However, they do not offer the designer to 1) test the classification and false-activation performance of body-based gestures and 2) modify and test gestures without re-recording them. In SequenceSense, I provide these to allow designers to identify potential issues with the

gestures and fix them without recollecting gesture samples.

2.2.2 Mitigating False Activations

As false activations have direct implications on the adoption of a particular technology that results in user frustration [23], several techniques were developed and tested. One of the most common approaches is to use *delimiter gestures*. By using delimiter gestures that are unique and different from regular body motion, the system may ignore any motions detected without the delimiter gesture [13, 139, 140, 141, 142]. A drawback of any delimiter action is the disruption in the user's workflow [143, 1]; the user must first perform the delimiter and then the intended gesture, therefore using the gestures less efficiently. Another approach for mitigating the false activation problem is to simulate the false activation performance using an everyday motion database [134, 134, 137]. While promising, these methods only support false activation checking against the whole gesture, so the designers cannot modify gestures in the design process, so they iterate through the process to make gestures usable. However, studies have shown that providing false activation analysis could help designers make gestures more usable in real deployment [144, 143, 145]. Inspired by these studies, I also use an everyday motion database for designers to easily identify gestures with potential conflicts with daily activities. Moreover, our method supports the analysis of the false activation against modified gestures, so the designers do not need to recollect data in the design process.

To summarize, designing good motion gestures involves important technical challenges, such as gesture classification performance and false activations, in addition to ensuring their usability. While the development of designer-friendly gesture design tools and the false activation simulation could accelerate the gesture design process, getting good motion gestures still requires multiple iterations of gesture design, data collection, and testing/analysis. SequenceSense aims to significantly reduce the effort of a gesture designer by eliminating the need for repetitive data collection when modifying gestures.

2.3 Making Gestures Compatible Across Devices

The recent advancement in technology and science has introduced numerous mobile and wearable devices equipped with lightweight sensors. These sensors have made it possible to establish new interaction prospects. These interactions can be categorized into several aspects, such as single-device to multi-device, multi-device to cross-device, cross-surface to multi-surface, cross-display to multi-display [146] interactions. Here, first, I discuss approaches that utilize single- and multi-device(s) for interactions. These interactions are mostly device-dependent, and they are not compatible across devices. Ultimately, I highlight how I can make a system like UnifiedSense to reduce device dependency in cross-device interactions and facilitate users to

reuse their learned gestures, promoting compatible gesture interactions. Finally, in this section, I discuss the state-of-the-art for recognizing gestures researchers explored over time and the choice of a state-of-the-art algorithm (i.e., Time-Series Transformer) to develop UnifiedSense.

2.3.1 Wearable and Cross-Device Interactions

Various commercial devices have emerged (e.g., Fitbit Versa [147], Oura Ring [148], and Google Glass [149]) and made it possible for researchers to create new interaction techniques using the device or for that device. For instance, wrist-worn and mobile devices can track people's physical activities (e.g., Fitbit Versa [147]) and perform daily tasks such as automatic voicemail playback (e.g., iPhone [150]). Wrist-worn devices also support various gestures such as pinch [151, 152, 153, 154] and hand gestures [155, 156, 157, 158, 159, 160, 161] for quick interactions. Similarly, finger-worn devices (e.g., Oura ring [148]) have been leveraged to detect interaction like click and drawing motions [162], detecting finger gestures [163], making a 3D signature [164], providing text input to computers [165], adjusting the lights in a home environment [166], controlling smart glasses using the thumb and middle fingers [167], scrolling for the computers [168], enabling in-pocket smartphone operation [169]. Eyewear enable users to blend digital content with the physical world [170], navigate content with eye blinks [171] or hand-to-face gestures [172], and control remote objects [173]. Many eyes-free interaction techniques have been developed to improve access to and awareness of the features offered by a mobile device, such as gesture-based authentication [174], and indirectly interacting with a phone with foot gestures while the phone is in the user's pocket [10]. Recently, the earbuds' microphone has been leveraged to detect touch gestures near ear positions (e.g., temple) and to interact with smartphones to access music players, phone calls, and notifications [37].

Prior work has also investigated novel interactions across multiple devices. Studies have revealed that users tended to allocate a complex task across multiple devices based on device form factors and functionalities [175, 176]. For example, Duet demonstrates various intriguing interactions, such as cross-device pinch gestures and touches made with a finger knuckle or a fingertip enabled by fusing sensor input across a watch and a phone [52]. Mayer et al. [177] proposed to combine the capabilities of head- and wrist-worn computers to interact with objects that are in the focus of a head-mounted display. Some research has shown cross-device sensing (e.g., sensing beyond itself [178]) for gesture recognition. EchoFlex [179] detects hand gesture using ultrasound imaging. Laput and Harrison presented hand activity detection using smartwatches [180]. TapID [181] indirectly senses touch interactions using wrist-worn wearable devices, though it is for more fine-grained finger identification beyond just sensing touches. Similarly, Ahn et al. [21] introduced an indirect text entry technique using a smartwatch for smart glasses. Furthermore, to enhance

robust interactions with cross-device modalities, some research has shown synchronous input signals by combing inputs from different devices [182, 63, 183, 184, 185, 142]. Besides, existing commercial products have started supporting cross-device behaviors such as using a watch as a remote to control the content on a phone [186] or automatic task continuation when the user moves from a phone to a computer [150].

Prior body-centric cross-device work has investigated interaction in coordination between two smart devices [52, 177], use of one smart device as input for another [21], and combining devices for synchronous input signals [182, 63, 183, 184, 185, 142]. Our motivation for designing UnifiedSense is somewhat different from these works. UnifiedSense assumes that users would wear multiple devices simultaneously. Subsequently, our system automatically learns gesture patterns by leveraging sensor data from available devices and the input on the primary device so that users can use gestures even without a primary device on them.

2.3.2 Wearable Gesture Recognition

For the last few decades, gestures have become a salient medium for natural interaction with computers, and various wearables have emerged to detect hand gestures. Here I present prior work on hand gesture recognition using wearable devices.

While some systems detect hand gestures using devices that are not worn on the arm or the hand, such as head-mounted devices (e.g., Meta Quest VR headset) and shoulder-mounted devices (e.g., FingerInput [187]), many use hand-worn and arm-worn devices that can detect the gestures at a closer distance. A wide range of sensors can be embedded or attached to a wristband or armband for hand gesture recognition, such as infrared (IR) ranging sensors [156, 188, 189, 190, 191], cameras [192, 193, 194, 195, 58], electromyography (EMG) sensors [196, 38, 13], acoustic sensors [14, 197, 157, 180], and stretch sensors [44]. The mostly used sensor for detecting hand gestures might be the Inertial Measurement Unit (IMU) [198, 199, 200, 8, 154, 201, 202], due to its ability to accurately and responsively detect motions, small sensor size, low cost, and being already deployed in many wearable devices possibly because of the aforementioned benefits.

Detecting hand gestures using hand- and arm-worn sensors is often done by using time-series analysis and machine-learning algorithms. For instance, several trajectory-based gesture recognition methods, such as dynamic time warping (DTW) [203, 198] and hidden Markov models (HMM) [204, 205], can recognize gesture trajectories (e.g., line, square, circle, star [205]) using few samples while achieving high accuracy. However, these methods may not work well for more complex and fine-grained gestures. More sophisticated techniques have emerged, relying heavily on data-driven approaches. For example, machine learning models such as support vector machines (SVM) and decision trees, e.g., [188, 197] are leveraged to design gesture recognizers. The gesture recognizers designed with traditional machine learning models do not scale well for a

high volume of data in case the model is deployed in the wild. Therefore, deep learning models have emerged for training a robust recognizer model which could be used in the wild, e.g., [192, 206, 202]. Motivated by these works, our approach also trains a high-performance Time-Series Transformer (TST) model [3, 2] from a moderate amount of data (collected from our user study).

In selecting the TST model for gesture recognition in UnifiedSense, I considered the reported performance and capabilities of various state-of-the-art algorithms, including TST, as described in the paper by Zerveas et al. [2]. The evaluation results presented in the aforementioned paper demonstrate that TST outperforms other methods, such as ROCKET [207], dilation-CNN [208], XGBoost [209], and LSTM [210], on a range of datasets, achieving the best average rank across multiple dimensions. Although our study primarily focused on simpler gestures, I chose the TST model due to its proven effectiveness in handling complex and fine-grained gestures. I acknowledge that the gestures used in our experiments may not be as intricate as those mentioned in the reasoning for algorithm selection. However, I believe that leveraging a powerful and versatile algorithm like TST allows us to establish a strong foundation for recognizing and understanding gestures in various contexts. Furthermore, our dataset consists of high-dimensional time-series data derived from multiple channels, with each channel comprising 8 tuples. Given the TST model's demonstrated efficacy in handling high-dimensional time-series data, it aligns well with our dataset characteristics, reinforcing our decision to employ TST in UnifiedSense. Although simpler approaches may yield better recognition rates for certain cases, our objective was to explore the potential of state-of-the-art machine learning technologies, such as transformers, in gesture recognition. I discuss the details of our approach in Section 4.1.1.

2.4 Investigation of Hand-Movements Manipulation

This section focuses on previous studies related to hand-movement manipulations, also known as perception manipulations, along with their advantages and limitations. It then moves on to how individuals develop motor skills through repetition, influenced by their previous experiences, and examines the potential negative impact of manipulations on this learning process when applied to hand movements. Additionally, it delves into the ways in which people adapt to these manipulations over time.

2.4.1 Perception Manipulation

Perception manipulation in VR refers to the technique that alters and controls an individual's perception of their own body and the virtual environment they are immersed in. It involves creating a sense of embodiment and ownership over a virtual body and manipulating sensory information to influence the perception of the

virtual environment [211, 212]. VR is a robust platform for employing perception manipulations, attributed to its capacity to create an illusion of non-mediation [213, 214, 215, 216].

A key application of perception manipulation is in retargeting user interactions to provide tangible haptic feedback from a limited set of physical objects. Techniques such as redirected touching and haptic retargeting allow users to feel physical contact in VR, despite the virtual nature of the objects they see [217, 218, 219]. This approach cleverly uses the visual-haptic discord to create realistic tactile sensations, enhancing the user's engagement with the virtual world. In addition to tactile feedback, VR perception manipulation extends to simulating weight sensations and scaling haptic feedback [220]. These methods adjust the perceived weight or size of virtual objects, enabling users to experience a more diverse range of interactions without the need for complex hardware setups. Such manipulations contribute to a richer, more varied virtual experience, offering new possibilities for VR applications beyond simple locomotion or visual immersion. Redirected walking is another significant aspect of perception manipulation, allowing for the exploration of large virtual spaces within physical confines [221, 222, 223, 224, 225]. This technique subtly adjusts the user's direction of movement, making rotational manipulations particularly effective due to their less noticeable nature compared to translational changes [226, 222].

Despite the innovative potential of these manipulation techniques, they could inadvertently affect the accuracy of skilled motor activities honed over repeated practice. Individuals depend on both visual and kinesthetic feedback to refine their movements, adjusting for discrepancies between intended and executed motions [227, 228, 229, 230]. The cultivation of such motor proficiency and muscle memory is essential for achieving task execution with greater velocity, consistency, and stability while minimizing the cognitive burden [231]. Nevertheless, the dynamic essence of perception manipulation, which alters the congruence between virtual and actual hand movements, introduces obstacles to achieving such automatism. Moreover, the scope of perception manipulation traverses beyond beneficial uses, intruding upon ethical dilemmas and hazards like cybersickness and potential exploitation, adversely affecting users' psychological and physiological well-being [35, 232]. For example, Casey et al. [34] uncovered a software flaw enabling the manipulation of VR safety boundaries, thereby orchestrating the "Human Joystick Attack" that secretively directs users' movements.

2.4.2 Motor Learning, Proprioception, and Kinematic Adaptation

Motor learning involves learning, retaining, and using physical skills. This process improves over time through practice and real-world use [233, 231]. Proprioception, on the other hand, is the internal sense that informs the body of its spatial position, accounting for limb and body movement. It is a vital component for effective

motor control [231]. These two things work closely together, especially when it comes to complex movements in VR [234, 235].

One well-known method in motor learning literature is classic visuomotor training. It uses two kinds of signals to teach. The first is error-based signals, which are differences between what humans intended to do and what actually happened. These are used for what's known as model-based learning [236, 237, 238, 239]. The second is visual-proprioceptive discrepancy signals, which are differences between where humans think of their hand is and where it actually is. These help humans adjust their movements on the fly [239, 240, 241, 242]. It is worth mentioning that humans can adapt very quickly to these signals, often within just six classic training trials [243, 244].

Another training method is *exposure training*, which refers to a training paradigm that involves the manipulation of sensory feedback to recalibrate proprioception and update predicted sensory consequences. This type of training typically involves a combination of robot-controlled hand movements with rotated visual feedback, which eliminates the ability to update predicted sensory consequences [245]. Despite the limitation of being used with robot-controlled hand movements, the method is just as effective for adjusting humans' sense of body position and movement [240, 241, 242, 246].

In this context, VR offers new opportunities and challenges for improving body position sense. The use of head-mounted displays (HMDs) can introduce visual-proprioceptive discrepancies, which may not only disrupt proprioception but also induce motion sickness[234]. Despite these disruptions, VR offers a convenient setting for dynamic manipulations, broadening the scope of motor learning to include not just sensorimotor adaptations but also knowledge-based action selection [247].

Artificial perturbations in perception, often delivered via distorted visual or tactile feedback, act as a catalyst for kinematic and proprioceptive adaptations. Such dynamic environments force the central nervous system to update its internal models, culminating in a robust and adaptive motor learning framework[235].

In summary, motor learning in dynamically perturbed environments constitutes a complex interplay between sensorimotor signals and cognitive strategies. While traditional motor learning predominantly depends on error-based and visual-proprioceptive discrepancy signals, emerging paradigms are emphasizing the significance of sensory recalibrations and knowledge-based action selection [239, 240, 241, 242, 247]. Moreover, the speed of these motor adjustments and proprioceptive recalibrations, often occurring within a few training trials, highlights the neural plasticity and adaptability of motor systems [243, 244].

This research narrows its focus to the investigation of motor performance metrics in VR environments subjected to perceptual manipulation. The study's primary aim centers on elucidating the trajectory changes in motor tasks as proficiency develops rather than identifying the contributory factors affecting such proficiency during manipulation.

2.5 Detecting Hand-Movements Manipulation

Hand-movement manipulations can be detected through various techniques. In this context, I have opted to use a specific deep-learning strategy known as the *anomaly detection approach*, utilizing hand-movement data to pinpoint manipulations. Subsequently, I will briefly discuss the anomaly detection method, prior works using the anomaly detection approach, and the rationale behind selecting this approach for our research.

Anomaly detection has significantly evolved since the 1960s, diversifying into areas such as surveillance [248], network quality [249, 250], and sensor systems for automated vehicles [251], driven by deep learning advancements. Deep learning's ability to analyze complex, high-dimensional, and temporal data has enhanced anomaly detection methodologies, particularly for temporal and sequential datasets.

Given the sequential nature of our hand-movement time-series data, I opted for Long Short-Term Memory Networks (LSTMs) [252, 253, 254]. LSTMs excel in handling sequential data through their Encoder-Decoder architecture, which processes variable-length input sequences to produce a comprehensive vector representation. This representation is then decoded to reconstruct the sequence, enabling precise anomaly detection in hand-movement data by learning speed and acceleration patterns.

Anomaly detection techniques are categorized into supervised, unsupervised, and semi-supervised models, with the choice largely dependent on data labeling and specific dataset challenges [255, 256, 257, 258]. Supervised models are accurate but constrained by label scarcity and imbalance issues. Unsupervised models, which use unlabeled data to detect patterns, can be compromised by noise [255]. Semi-supervised models utilize available normal data labels to improve outlier detection, offering an effective compromise by leveraging the relative ease of obtaining normal data over anomalous labels [259]. This approach is notably effective in network traffic analysis, where it outperforms unsupervised methods [255]. Inspired by these findings and given the challenges associated with gathering manipulated labeled data for hand movements in VR environments, I have chosen to employ a semi-supervised learning approach. This method allows our model to be trained to recognize the patterns of normal movements, effectively addressing the constraints of our specific application context.

In summary, ManipulaSense's adoption of the LSTM model is strategically chosen to leverage its proficiency in sequential data analysis, which is essential for the nuanced understanding of hand-movement time-series data. This decision aligns with the broader trend in anomaly detection towards utilizing deep learning models capable of sophisticated temporal data analysis.

Chapter 3

Designing Conflict-Free Body-Based Gestures

Designing and implementing usable body-based gestures is challenging as regular body movements (unintentional) can easily be interpreted as gestures and cause false activations [260]. Adjusting the gesture recognizer or modifying gestures to cause fewer false activations may lead to false negative errors, where the system does not recognize intended gestures. Furthermore, only focusing on the gesture recognition performance may result in a gesture with poor usability. To make gesture systems good at preventing false positive and false negative errors, designers "have to get down and dirty with your recognizer, and manually tweak your gestures using the tools and methods ..." [23]. This process often involves validating recognizer performances through human subject studies [30, 31] in both lab and real settings that makes the process time-consuming and expensive, resulting in difficulties in efficiently iterating on the gesture design.

Several methods have been developed to ease the process of designing usable body- and motion-based gestures [36, 1]. For example, Parnami et al. developed Mogeste [36], a mobile tool to allow gesture designers to design, modify, and test motion gestures anywhere. By letting gesture designers be at the site where the gestures will be used and test and modify the gestures in a realistic scenario, Mogeste can help them identify potential gesture conflicts and accelerate the gesture design process. However, it still requires the designers to test the gestures in a realistic scenario to evaluate the usability, and modifying gestures needs re-collection of gesture samples to train the recognizer. On the other hand, MAGIC [1] allows gesture designers to quickly gain insights into the designed gestures' performance, recognition performance, and potential conflicts. A designer can design a set of gestures on MAGIC and record gesture samples, similarly to Mogeste [36]. The recorded samples are used to train a gesture recognizer, which provides inter- and intra-class classification

performances. In addition, MAGIC uses an everyday hand motion repository to estimate potential conflicts so that the designer does not have to run a user study to test its usability. However, once a gesture with potential conflicts is identified, the gesture designer should modify the gesture and re-collect the gesture samples to check the usability of the new gesture, which can be costly, especially if multiple iterations are required (Figure 3.1).

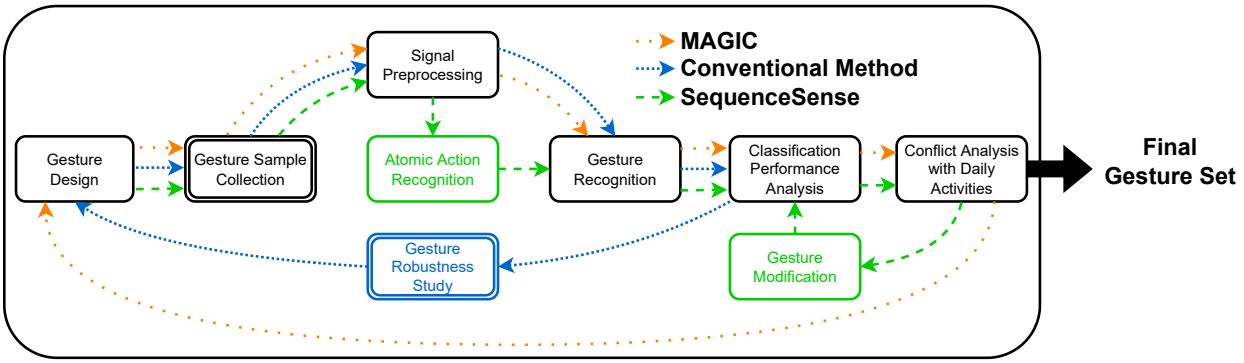


Figure 3.1: This figure depicts the difference between the conventional method of usable gesture design, MAGIC [1], and SequenceSense. Using the conventional method, the designer has to perform multiple user studies (double-border boxes) for gesture data collection and performance evaluation in real settings to design a usable gesture set. MAGIC [1] compares the gestures with the regular movements in the loop of designing gestures to avoid false activations; it does not automate finding the usable gesture set if false activation occurs with the selected gesture set. With SequenceSense, designers do not need to redesign the gesture and re-collect gesture samples; rather, they can modify the atomic action sequence of conflicting gestures.

This chapter presents SequenceSense, a tool developed to help designers efficiently build a usable body-based gesture set using a sequence-based gesture recognizer and an automatic conflict analyzer. Instead of training a gesture recognizer using raw features, SequenceSense detects atomic actions in a gesture and uses their sequence to define and recognize gestures. This allows the gesture designer to easily modify the gestures by reconfiguring atomic actions in the gesture sequence and to immediately check the recognition performance and potential false activations without re-collecting the gesture samples or conducting experiments (Figure 3.1). I believe that this will allow gesture designers to focus more on the usability of gestures and less on their robustness.

SequenceSense has three main components: recognizer, analyzer, and editor (Figure 3.2). The gesture recognizer segments gesture samples into smaller movements and classifies them as atomic actions so that a gesture can be represented as a sequence of atomic actions (e.g., a foot tap to the right: foot lift, move to the right, foot land with tapping). The use of a sequence-based gesture recognizer can help designers find and modify atomic actions that cause conflicts with other gestures and/or daily activities. The gesture analyzer compares gesture sequences against each other and estimates the possible conflicts between gestures and false activations during daily activities using a database containing atomic actions collected during daily activities.

The gesture editor is the front-end of the system where the designer can check recognition performance, identify possible conflicts, and modify gestures.

To evaluate the feasibility of SequenceSense, I chose foot gestures, which are shown to be preferred over other body-based gestures in scenarios where users cannot use their hands [9]. I adopted 17 foot gestures (Table 3.2) from prior work on body-based and foot gestures [25, 261, 262, 263] that focused on usability.

I conducted a user study to validate our tool in effectively designing gestures and compare its effectiveness over MAGIC [1]. I measured the quality of the designed gestures, the efficiency of designing gestures, and the overall user experience. The results from our user study showed that using SequenceSense, participants were able to design usable gestures with shorter gesture sequences, requiring approximately half of the time compared to MAGIC. Moreover, all the participants in our study preferred using SequenceSense over MAGIC to design gestures as gesture design with SequenceSense requires less effort and especially does not require participants to re-collect modified gesture samples when conflicts occur.

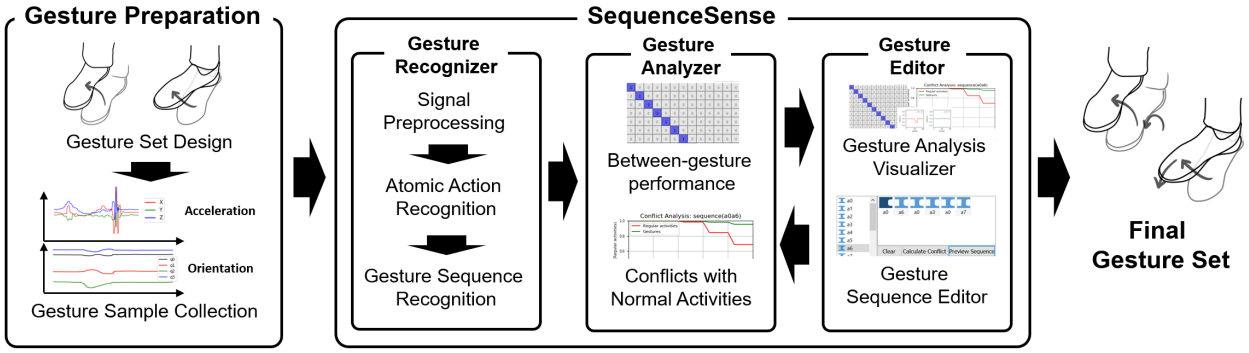


Figure 3.2: Designing usable gestures using SequenceSense. Once gestures are designed and recorded, SequenceSense 1) segments and classifies atomic actions forming the gestures, 2) finds the sequence of atomic actions for each gesture, and 3) analyzes gesture classification performances and possible conflicts with daily activities. With the visualizations and the gesture sequence editor in SequenceSense, the designer can modify the gestures to be more usable.

3.1 SequenceSense

SequenceSense is a gesture designing tool that allows designers to efficiently design, test, analyze, and modify usable foot-based gestures. While there exists several tools for designing and analyzing gestures [1, 36, 264, 138], the highlight of SequenceSense is its gesture design process which eliminates the need for multiple data collection studies to evaluate the gestures' usability, and instead requires only the initial gesture sample collection (Figure 3.2). This enables the designers to make gesture modifications without requiring them to re-collect the modified gesture data. SequenceSense achieves this by representing gestures in terms of their primitive movement units, *atomic actions*, which the gesture designer can use to compose new gestures by

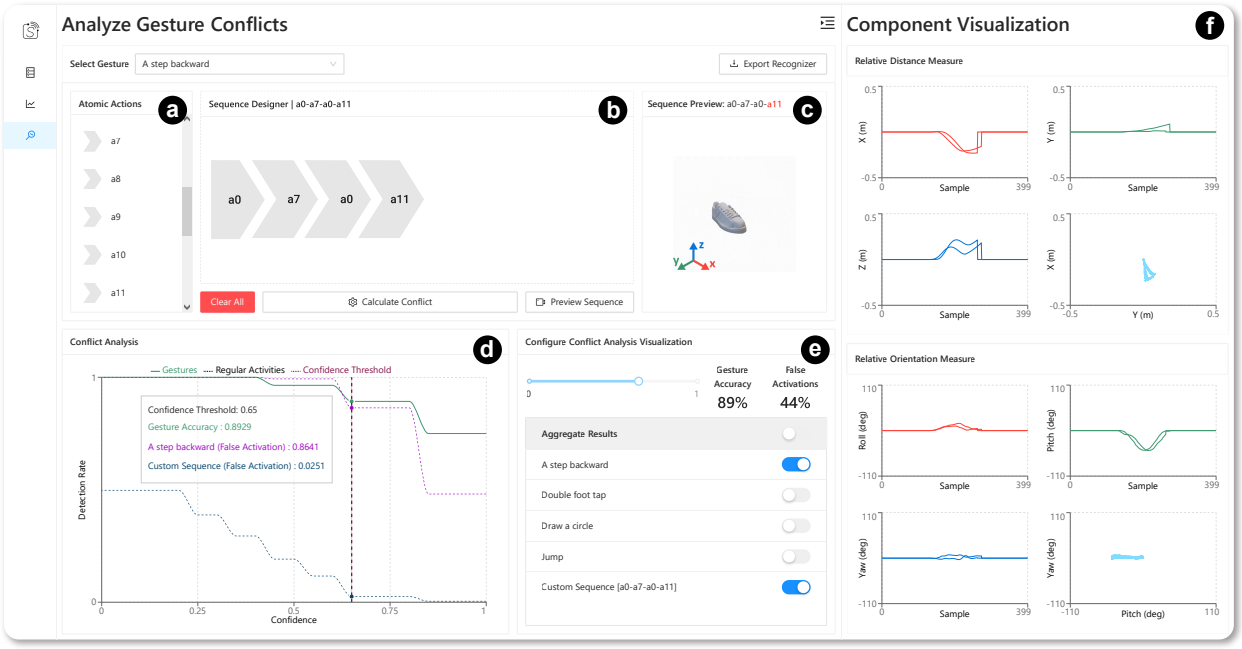


Figure 3.3: SequenceSense Gesture Designer Tool. The gesture *a step backward* has a very high false activation detection rate of 0.86 for a confidence threshold of 0.65. Using SequenceSense, I can redesign that gesture by appending a foot tap to it, consequently reducing the overall false activation detection rate to 0.03.

sequencing these atomic actions in various combinations. This technique can be especially helpful in creating usable gestures with low false activations since it eliminates the bottleneck of re-collecting gesture samples, facilitating rapid iterations.

I implemented SequenceSense as a web application using approximately 5,000 lines of JavaScript (ReactJS) and Python code (Figure 3.3). For our implementation, I focused on the foot-based gesture design scenario. I chose 17 foot gestures (listed in Table 3.2), the first 14 of which were taken from the elicitation study by Felberbaum and Lanir [25], and the last three were complemented from the user-defined elicitation studies on foot gestures and interactions [261, 262, 263], as the baseline gesture set.

Designing Gestures Using SequenceSense

The workflow of designing usable gestures with minimal false activation using SequenceSense is described below.

Step 1: Classifying Gesture Samples

The designer first collects samples of foot gestures and uploads them into SequenceSense. SequenceSense supports both *labeled* and *unlabeled* gesture samples in CSV format containing 3-axis acceleration and

orientation represented as quaternions. Once the designer has uploaded all the gesture samples, SequenceSense classifies the samples and shows the gesture recognition accuracy through a confusion matrix.

Step 2: Analyzing Gesture Conflicts

SequenceSense compares the uploaded gesture samples against a repository of daily foot activities and reports the overall gesture conflicts (Figure 3.3d). The detection rates of these conflicts are visualized along with the corresponding gesture accuracies for confidence values ranging from 0 to 1. The designer can now adjust the confidence threshold and observe the classifier performance in terms of gesture accuracy and false activations for the selected confidence value (Figure 3.3e).

Step 3: Visualizing Gesture Components

If the false activation detection rates of the uploaded gesture samples are high, SequenceSense allows the designer to break down the gesture samples into their individual acceleration and orientation components and visualize them separately to intuitively identify the cause of conflict (Figure 3.3f). Based on their observations, the designer can plan gesture modifications accordingly to avoid false activations.

Step 4: Redesigning Gestures

Once the designer has planned their gesture modification, they can use the *Sequence Designer* panel (Figure 3.3b) provided by SequenceSense to compose the new gesture through a sequence of atomic actions (Figure 3.3a). SequenceSense also allows the designer to visualize the redesigned gesture through animation (Figure 3.3c). After redesigning the gesture, the designer can check for conflicts by simply clicking on the *Calculate Conflict* button. In this way, the designer can iteratively design gestures and evaluate their corresponding conflicts until they are satisfied with the usability of the redesigned gesture, without having to manually perform or collect these gestures.

Step 5: Exporting Recognizer

Finally, once a usable gesture is designed, the designer can export the gesture recognizer model along with the associated configurations to deploy in their machine learning pipeline.

3.2 Gesture Recognition

In this section, I discuss the device used for data collection, data preprocessing, feature sets, atomic action detection, and the sequence-based gesture recognizer that SequenceSense uses.

3.2.1 Device for Data Collection

I designed a wearable device to collect foot movement data, as illustrated in Figure 3.4. The device was designed to be attached on top of a shoe. The device comprises a $SAMD21$ microcontroller, a $BNO080$ 9-DOF IMU, a micro-SD card, and a 3.7 V, 900 mAh Lithium polymer battery in a $3D$-printed case ($52 \times 34 \times 22$ mm). The device collects acceleration and orientation with sampling rates of 200 Hz.

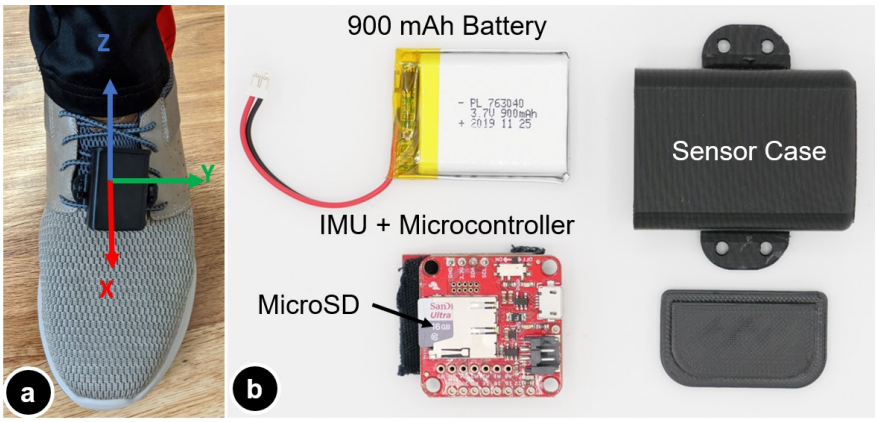


Figure 3.4: (a) The data collection device worn on the shoe and (b) components in the device.

3.2.2 Data Preprocessing

SequenceSense uses the time-series data from the device's IMU sensor which includes the timestamp t, acceleration ($a(t)$), and device orientation represented in quaternion ($Q(t)$), which is then denoised using Kalman filter [265]. The data preprocessing pipeline ensures that the position and orientation are aligned with the user's foot (Figure 3.5). Our technique for representing the collected sensor data in the local body frame is inspired by the method used by Ruoyu [266]. Once the sensor data is aligned with the foot orientation, our system segments the data into stationary and non-stationary parts using the *zero velocity update* (ZUPT) algorithms [267, 268]. I derive distance from acceleration using double integration (from acceleration to velocity and then to distance). The noise accumulated by this method is offset by the ZUPT algorithm. Finally, I derive relative orientation in Euler angles from relative quaternions [269] as follows. First, the relative quaternion is calculated for the zero velocity update portion as $Q_r(t) = Q_{s_{t-1}} * Q'_t$, where $Q_{s_{t-1}}$ is the quaternion of the previous stationary period at time $t-1$, and Q'_t is the conjugate quaternion of Q_t. Second, for the velocity, $v(t) \neq 0$, the relative quaternion $Q_r(t) = Q_t * Q'_t$ is calculated. Finally, the quaternion to *Euler transformation method* is employed to calculate $Roll(t)$, $Pitch(t)$, and $Yaw(t)$ angles from the combined $Q_r(t)$.

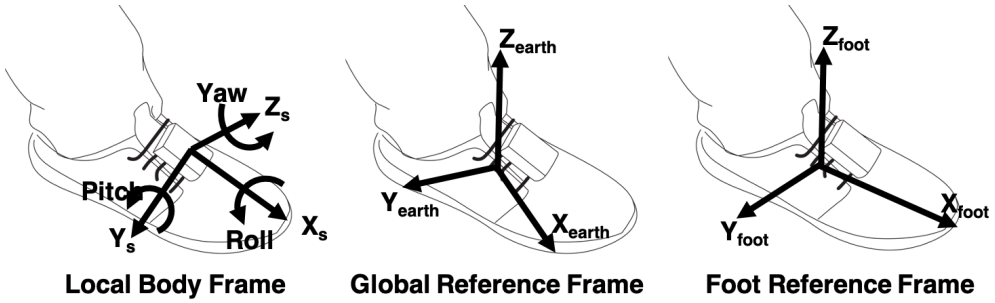


Figure 3.5: Local body frame, global reference frame, and foot reference frame.

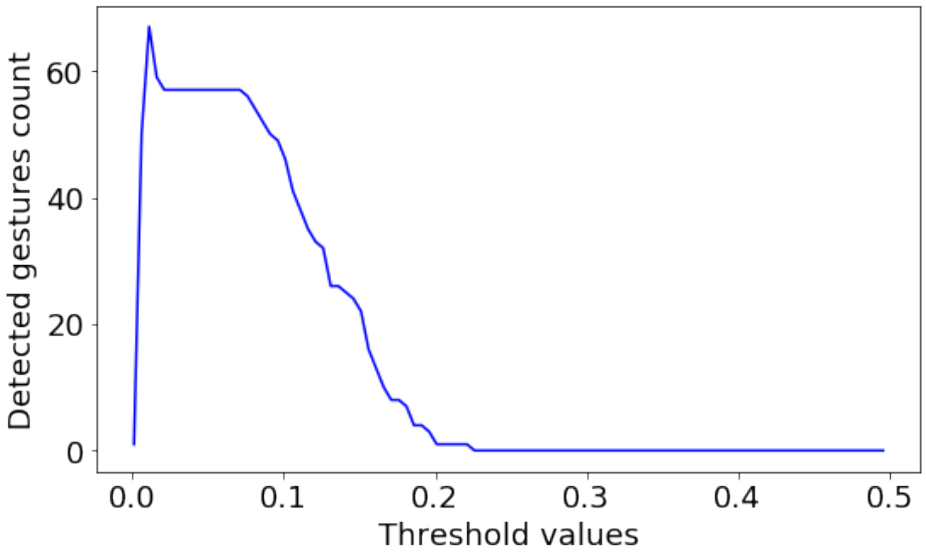


Figure 3.6: Number of gestures for different non-stationary detection threshold a_{th} values. I chose a_{th} of 0.03 because it is in the middle of the plateau, and using the value, our system segmented the desired number of the non-stationary segments.

Recall the segmentation of the stationary and non-stationary parts mentioned above, which is done using a threshold. I named this threshold to *non-stationary detection threshold*, (a_{th}). The threshold is calculated by aggregating 57 segmented *a step forward* gestures. These 57 gestures are arbitrary time-length stationary period readings of acceleration and orientation. The same zero-velocity update algorithm was employed to detect the stationary and non-stationary portions. Then the number of the different non-stationary portions is measured for the different values of a_{th} ranging from 0 to 0.5 (see Figure 3.6). Using a mathematical phenomenon widely used for threshold detection named *plateau* (between 0.0 and 0.1 in Figure 3.6), the threshold is found to be **0.03** in our case.

3.2.3 Feature Set

The foot movement kinetics can be determined using distance and orientation measures. For example, the Z-axis distance measure can correspond to the lifting and landing of the foot. Similarly, the pitch orientation measure is affected by the lifting and lowering of the toes. I thus use a dynamic-length 6-dimensional feature set, similar to a trajectory-based dynamic hand-gesture-pattern representation system [270]. These features are the distance measures in meters towards X, Y, and Z directions and angle measures in degrees corresponding to roll, pitch, and yaw angles. After analyzing the features, I discovered that the Z-axis distance measure follows a rising and falling pattern for all gestures except for dragging gestures and drawing a circle. For the *rotate toes left and right* gestures, the Z-axis distance measure is barely noticeable, but the pitch angle indicates that the foot is lifting and lowering. Both the Z-axis distance measure and pitch angle remain silent during dragging gestures and drawing a circle. The *walk in place* and *run in place* gestures exhibit the same repeating pattern. A *step to the side* gesture has a significant positive Y-axis distance measure, while a *step forward* and *step backward* gestures have a large positive and negative X-axis distance measure, respectively. The left and right turn gestures can be understood by observing the yaw angles, which reflect each other. Observing these features allows for easy and intuitive identification of gesture patterns. Other interesting observations are left for the reader to discover. A complete list of features illustrating the 17 gestures is reported in Figure 3.14a - Figure 3.14q.

3.2.4 Atomic Action

I define *atomic actions* as the building blocks of gestures. Atomic actions are small, primitive movements, such as lifting the foot up, moving the foot to the right, and placing the foot down. For example, a *foot tap* gesture can be constructed using two atomic actions – lifting the foot up and placing the foot down. This process of compositionally designing gestures using primitives has been extensively explored in prior literature and has yielded promising results in terms of increasing the accuracy or reducing the training set [271, 272, 273, 274, 275].

To detect atomic actions, our system segments the gestures based on the Z-axis distance measure and/or pitch angle. These two features are used to segment gestures because they provide insight into whether the foot is lifted from the ground. Gestures with both of these features silent (barely noticeable) are considered to be atomic action as a whole. Whether the Z-axis distance measure and pitch angle are silent or not is determined based on the slope. The slope is calculated by detecting the peak of the Z-axis distance measure and pitch angle. If the absolute positive slope is less than a threshold value, then both the distance and angle measures are considered silent, and the gesture is considered an atomic action. I found that the threshold

Table 3.1: Atomic actions explained with 6 positional and rotational features. The following symbols represent the features' pattern: "↗": increasing, "↘" : decreasing, "→": flat or silent, "↻": clockwise circular, "d": don't care.

Atomic action	Interpretation	X	Y	Z	Roll	Pitch	Yaw
a_0	Foot/Toes lifting	d	d	↗	d	↗	d
a_1	Foot down (while jumping)	d	d	↘	d	↗	d
a_2	Foot down (while running in place)	↘	↘	↘	↘	↘	↘
a_3	Foot down (while stepping to right side)	→	↗	↘	d	↗	↘
a_4	Foot down (while turning left)	d	↘	↘	d	d	↘
a_5	Foot down (while turning right)	d	↗	↘	d	d	↗
a_6	Foot down (while stepping forward)	↗	→	↘	d	↗	d
a_7	Foot down (while stepping backward)	↘	→	↘	d	↗	d
a_8	Foot down (while walking in place)	→	d	↘	↗	→	→
a_9	Toes down (while rotating toes left)	→	↘	↘	↘	↘	↘
a_{10}	Toes down (while rotating toes right)	→	↗	↘	↗	↘	↗
a_{11}	Toes down (while tapping)	d	d	↘	→	↘	→
a_{12}	Toes down (while tapping to the right)	d	d	↘	↗	↘	↗
a_{13}	Toes down (while tapping to the left)	d	d	↘	↘	↘	↘
a_{w_1}	Foot drag (while drawing a circle)	↻	↻	→	d	→	d
a_{w_2}	Foot drag (while dragging from front to back)	↗	d	→	d	→	d
a_{w_3}	Foot drag (while dragging from left to right)	d	↗	→	↗	→	↗
a_{w_4}	Foot drag (while dragging from right to left)	d	↘	→	↘	→	↘

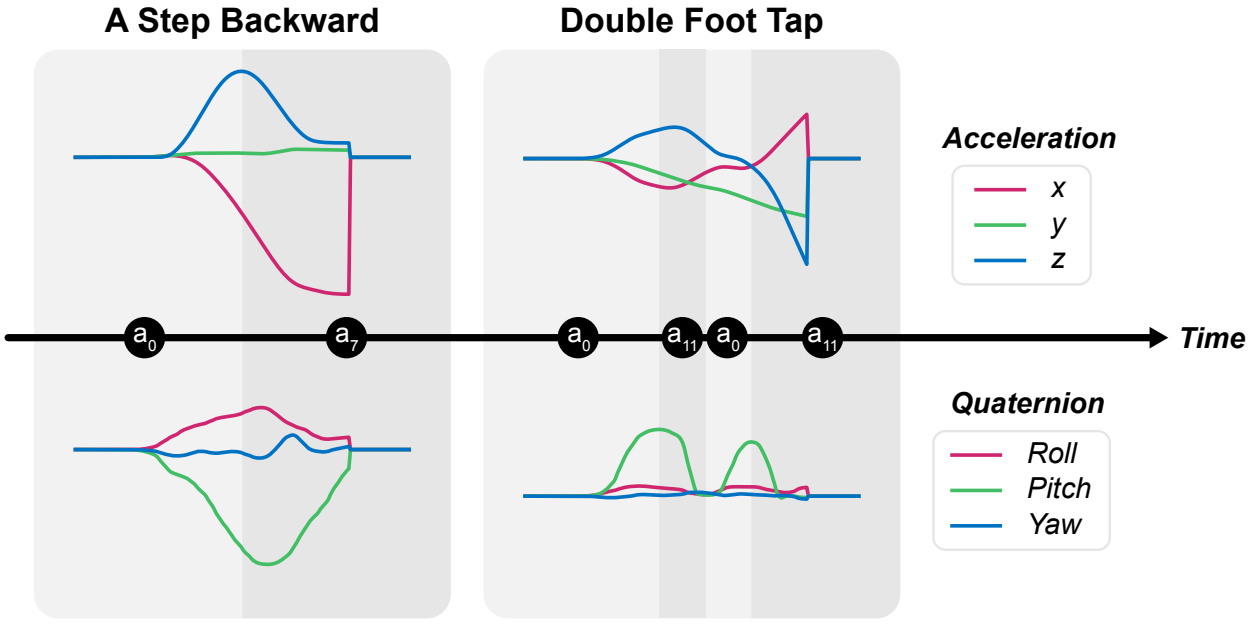


Figure 3.7: Illustrative examples of the segmentation process. *A step backward* gesture is segmented into two atomic actions (a_0a_7) (left side). *Double foot tap* gesture is segmented into four atomic actions ($a_0a_{11}a_0a_{11}$) (right side).

slope for both the Z-axis distance measure and pitch angle is 8 degrees. For non-silent gestures, our system detects a pattern of increasing (moving forward/rightward/upward) and decreasing with a positive peak value for the Z distance measure, followed by the same pattern for the pitch angle. Our system then segments the gesture into two parts: from the zero crossing to the next positive peak as the first segment and from the peak to the next zero crossing as the second segment. This process is repeated until the entire gesture has been segmented. Two examples are illustrated in Figure 3.7, showing the segmentation of *a step backward* and *double foot tap* gestures, respectively.

Some atomic actions are more complex than others due to a lack of sudden changes in the atomic action. For example, the dragging and drawing circle gestures are classified as single atomic action gestures. The *run in place*, *walk in place*, *double foot tap*, *foot tap to the right*, and *foot tap to the left* gestures are segmented into four atomic actions. The first and third segments and the second and fourth segments are the same for these gestures. Therefore, the number of atomic segments is halved for these gestures. However, for the *foot tap to the right* and *foot tap to the left* gestures, the first and third segments are the same but not the second and fourth. The third segment of *foot tap to the right* is the same as the fourth segment of *foot tap to the left* and vice versa because the gestures are mirrored versions of each other. Except for these gestures, all other gestures (Indices 1, 3-7, 11, 12 in Table 3.2) are segmented into two segments, and the last segment clearly describes each gesture individually.

I identified 18 unique atomic actions in our system, which I obtained by segmenting the 17 gestures in our gesture set. Thirteen gestures from our gesture set start with lifting the foot up, which is defined by the atomic action a_0. Subsequently, these gestures are concluded by atomic actions a_1 through a_{13}. The remaining four gestures from our gesture set do not involve foot lifting and instead require the dragging of the foot. These gestures are considered atomic actions as a whole and are represented by atomic actions a_{w_1} through a_{w_4}. Table 3.1 reports the identified atomic actions, which are represented based on the 6-dimensional feature set. Finally, I admit that the 18 atomic actions defined here are not representative of all foot gestures and only represent the 17 gestures that I selected for our system. The breakdown of each gesture into atomic actions is illustrated in Table 3.2.

3.2.5 Classification

I implemented an atomic action classifier using the k-nearest neighbors (k-NN) [276] algorithm. The input dimension of the classifier was 400 samples (2 *secs*). I empirically found $k = 5$ to be ideal for our classifier. Since I are using time-series features such as distance and orientation measures, I used the Dynamic Time Warping (DTW) [277] algorithm to calculate the distance metric for the k-NN classifier.

Table 3.2: The chosen gesture sets and their definition by sequencing atomic actions.

Index	Gestures	Sequence of atomic actions
1	Jump	a_0a_1
2	Run in place	$a_0a_2a_0a_2$
3	A step to the side	a_0a_3
4	Turn left	a_0a_4
5	Turn right	a_0a_5
6	A step forward	a_0a_6
7	A step backward	a_0a_7
8	Walk in place	$a_0a_8a_0a_8$
9	Draw a circle	a_{w_1}
10	Drag from front to back	a_{w_2}
11	Rotate toes left	a_0a_9
12	Rotate toes right	a_0a_{10}
13	Drag from left to right	a_{w_3}
14	Drag from right to left	a_{w_4}
15	Double foot tap	$a_0a_{11}a_0a_{11}$
16	Foot tap to the right	$a_0a_{12}a_0a_{13}$
17	Foot tap to the left	$a_0a_{13}a_0a_{12}$

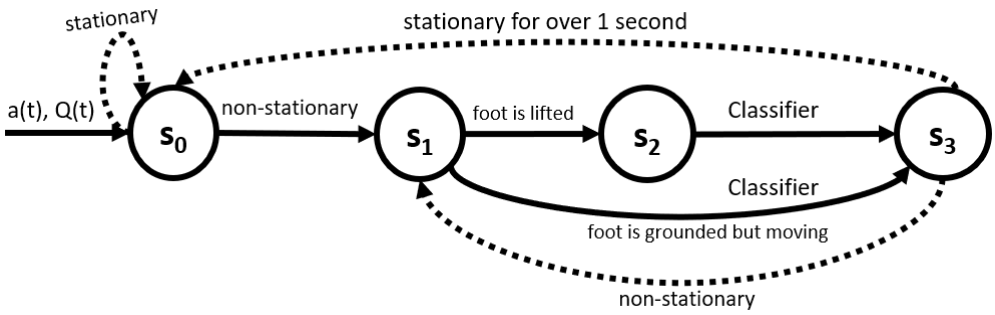


Figure 3.8: A state-machine-based gesture recognizer.

3.2.6 Sequence-Based Gesture Recognizer

I implemented a sequence-based gesture recognizer using a finite state machine which has shown to be effective in modeling human gestures [278]. Figure 3.8 represents the gesture recognizer based on the atomic actions detected (Table 3.2). Our sequenced-based gesture recognizer consists of four states. The initial state (s_0) is the stationary state, implying that the state-machine is in this state if there is no movement at all. Once the non-stationarity of the device is detected, the state-machine reaches the next state (s_1). Being in the state s_1 means that there may be a gesture to be detected. Next, the state-machine determines whether the

movement is done by lifting or dragging the foot. If movement is done by dragging the foot, the state-machine segments the whole movement into an atomic action by classifying it into one of $a_{w_1} - a_{w_4}$, and then reaches to the final state (s_3). Otherwise, if the state-machine detects the movement done by lifting the foot, then it reaches an intermediate state (s_2) by detecting the initial part of the gesture, i.e., a_0. In state s_2, the machine segments the remaining foot movements, and then classifies them into one of the atomic actions ($a_1 - a_{13}$), finally reaching state s_3. The final state s_3 is responsible for constructing complete gestures from previously classified atomic actions. If the stationarity of the foot movement in this state is over 1 second, then the machine outputs the sequenced gesture and resets it to the initial state s_0. Otherwise, if there exists foot movement within 1 second, the machine returns to state s_1 and continues to detect additional atomic actions.

3.3 Gesture Recognizer Performance Analysis

In this section, I evaluate the performance of our gesture recognizer in terms of its recognition accuracy and its ability to reject false activations.

3.3.1 Data Collection

I conducted a data collection study to collect foot movement data. The study was divided into two phases – (1) collecting natural foot movement during daily activities, and (2) collecting foot movement while performing the aforementioned 17-foot gestures (Table 3.2). The study was approved by the University of Virginia–IRB (#3446) and conducted while following the guidelines provided by the public and institutional health guidelines.

Participants

I recruited 12 participants (6 female, 6 male) from our university through email groups and word of mouth, with participants' ages ranging between 23 and 34 years ($M = 28.75$, $SD = 3.09$). All participants were right-footed, and they wore the device on their right foot. Participants were compensated with 50 USD for their time.

Phase 1: Daily Activities

The purpose of the first phase was to collect foot movement during daily activities. In this phase, participants were asked to wear the wearable device on their shoes for at least 6 hours a day, for two days. The participants performed their usual daily activities while wearing the device. In total, the daily activities dataset contains about 160 hours of data from 12 participants.

Phase 2: Base Foot Gestures

In this phase, I collected the foot movement while the participants performed the foot gestures (Table 3.2). After the researcher demonstrated the gestures to the participants, they were asked to perform each of the gestures in front of the researchers while wearing the device. They performed each of the gestures 10 times while being seated and standing, individually. For the seated condition, four gestures (*turn left*, *turn right*, *run in place*, and *a step forward*) were not performed due to the difficulties associated with performing them while being seated. In total, this phase resulted in the collection of 2040 standing gestures (17 *gestures* $\times$ 10 *samples* $\times$ 12 *participants*) and 1560 sitting gestures (13 *gestures* $\times$ 10 *samples* $\times$ 12 *participants*), and took about 50 minutes for each participant to complete.

3.3.2 Gesture Performance

I performed our gesture performance evaluations using leave-one-participant-out cross-validation. Our state-machine-based recognizer achieved a high accuracy of 97% in classifying 17 gestures. From Table 3.3, I observe high recall and F1-scores for all gestures except *jump* and *run in place*. These aberrations can be attributed to the undirected foot orientations that may occur while performing such gestures, and distance/orientation measures may not be sufficient in capturing these nuances. Figure 3.9 illustrates the confusion matrix for the 17 gestures.

3.3.3 Conflict Analysis

Natural body movements while performing daily activities may inadvertently trigger false gesture activations [4, 23]. I used our state-machine-based recognizer to identify all the falsely triggered gestures from the daily activities data that I collected in phase 1, and subsequently used it to detect the overall conflict of our 17 predefined gestures. Figure 3.10a illustrates the conflicts analysis for all of the 17 predefined gestures. Here, I can observe the overall false positive rate to be 58.5% for a confidence threshold of 0.85. Gestures with four highest false positive rates were identified as – *a step forward* (18%), *turn left* (16%), *turn right* (14%), and *a step backward* (12%).

3.3.4 Modifying Gestures to Reduce False Activations

In our conflict analysis, I observed a significant amount of false activations for most of the 17 predefined gestures. While these gestures may individually have a high probability of occurrence during daily activities, sequencing these gestures along with other gestures can potentially reduce their occurrence probability since it is unlikely for multiple gestures to occur within a short period of time. I thus modified all 17 foot gestures

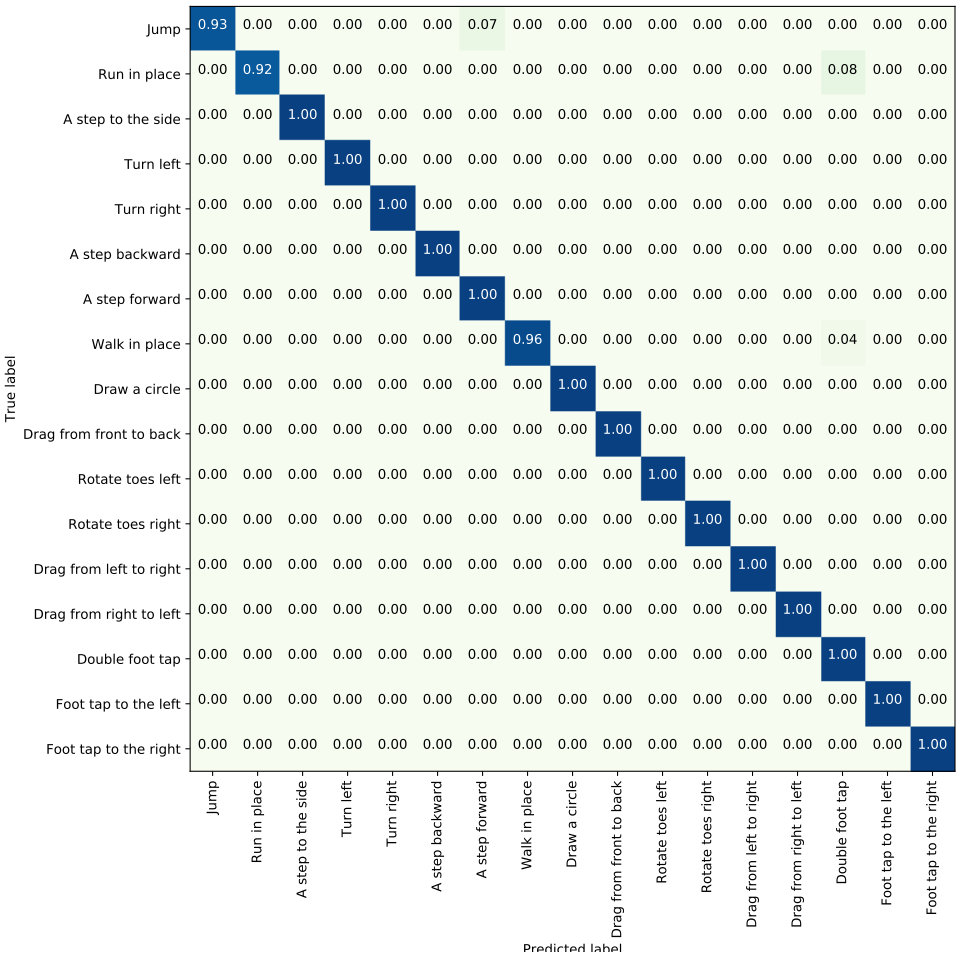


Figure 3.9: Confusion Matrix (CM)

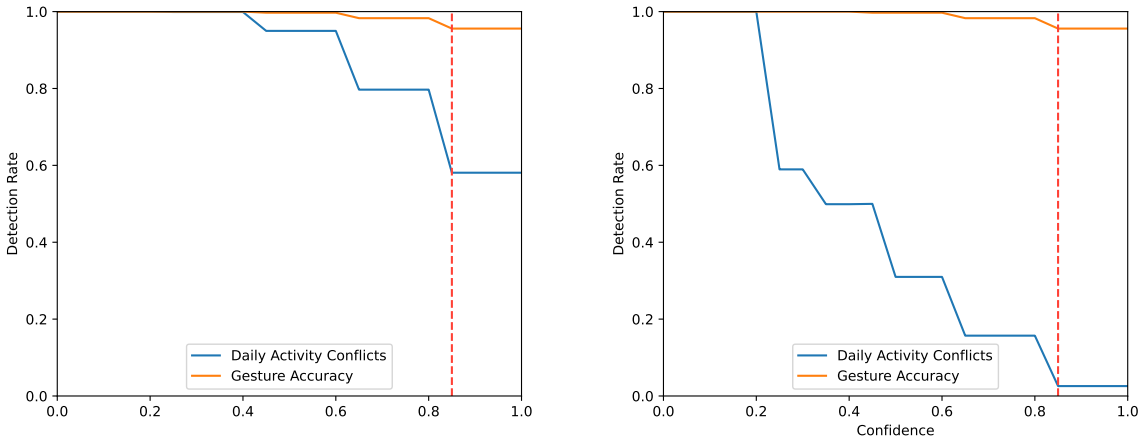
by prepending the atomic actions corresponding to a foot tap ($a_0 a_{11}$) to each of the original gestures using SequenceSense. Running the conflict analysis procedure again on these modified gestures yielded a false positive rate of 2.6% for the same confidence threshold of 0.85. The conflicts between both the modified and unmodified gestures are illustrated in Figure 3.10.

3.4 Tool Evaluation

I conducted a user study to determine the effectiveness of SequenceSense in allowing gesture designers to design usable gestures with low false positives. In particular, I compared our method against the MAGIC approach [1] to measure the efficiency of designing gestures, the quality of the designed gestures, and the overall user experience of using the atomic action sequence editor.

Table 3.3: Precision, recall, and F1-score of the gesture recognizer.

Index	Gestures	Precision	Recall	F1-score
1	Jump	1	0.93	0.97
2	Run in place	1	0.92	0.96
3	A step to the side	1	1	1
4	Turn left	1	1	1
5	Turn right	1	1	1
6	A step backward	1	1	1
7	A step forward	0.94	1	0.97
8	Walk in place	1	0.96	0.98
9	Draw a circle	1	1	1
10	Drag from front to back	1	1	1
11	Rotate toes left	1	1	1
12	Rotate toes right	1	1	1
13	Drag from left to right	1	1	1
14	Drag from right to left	1	1	1
15	Double foot tap	0.95	1	0.97
16	Foot tap to the left	1	1	1
17	Foot tap to the right	1	1	1



(a) Unmodified Gesture.

(b) Modified Gesture.

Figure 3.10: Comparison of conflicts between unmodified and modified gestures using SequenceSense. Red lines indicate the trade-off region between true positive (TP) rate and false positive (FP) rate while choosing confidence there.

3.4.1 Study Design

I designed a 2×2 within-subject study of designing 2 sets of gestures (each containing 3 unique gestures) using 2 methods – (1) MAGIC [1], and (2) SequenceSense. The MAGIC condition uses a modified SequenceSense tool that does not have the sequence editor so that the workflow of designing gestures involves the re-collection of gesture samples after gesture modification, as in its original design [1].

3.4.2 Procedure

The participants were asked to design three usable gestures unique to each method. For our study, I defined a usable gesture as a gesture having over 90% recognition accuracy and less than 10% false activations from daily activities. I chose 10% false activations because I found from the experimental analysis that the number of false activations is approximately nine conflicts/hour on average if I set the false activations percentage to 10%. This analysis was done on the collected daily activities database. The order of methods was counterbalanced to minimize the order effect.

The participants were required to attach the motion data collection device (Figure 3.4) on the shoe of their dominant foot for recording gesture samples. Before designing gestures using each method, the experimenters gave a brief demonstration of performing the gestures and designing gestures using the method. For each method, the participants were first asked to record one sample for each unmodified gesture from the gesture set. After recording the samples, the experimenters collected the gesture data from the device and provided it to the participants. The participants then imported the gesture samples into the tool to check their overall recognition accuracy and false activation. If the recognition accuracy and false activation of any gesture did not meet our predefined criteria, the participants had to modify that gesture until it satisfied the criteria. The participants were given a maximum of 30 minutes to design and modify the three gestures for each method. At the end of each method, the participants were asked to complete a survey on their experience of designing gestures using that method. Additionally, at the end of the study, the participants were asked to complete a final survey about their experiences of using both methods.

3.4.3 Participants

I recruited 9 participants (1 female, 8 male) with participant ages ranging between 24 and 32 years ($M = 28.5$, $SD = 3.5$). Participants were required to have prior experience in gesture design. All the recruited participants were graduate students from our university, and were right-footed. Participants were asked to rate their experiences in (1) using motion-based gestures, (2) implementing user interfaces, and (3) designing gestures using a 5-point Likert scale, with 1 being *novice* and 5 being *expert*. On average, the participants rated

Table 3.4: Quality and efficiency of designing gestures using MAGIC and SequenceSense (SS).

Gesture Set	Gestures	Average Accuracy (%)		Average False Positive (%)		Average no. of Iteration		Average Seq. Length	
		MAGIC	SS	MAGIC	SS	MAGIC	SS	MAGIC	SS
1	A step to the side	100	97.5	3.5	2	1.25	1	5.5	4
	Turn left	100	98	0.6	2.5	2.2	1	5.8	4
	A step backward	100	97.5	0.75	2	1.75	1	6.5	3.75
2	Turn right	100	98.4	3.3	2.4	1.3	1.2	5.3	4.6
	Rotate toes right	96.7	98.8	4.3	1.4	3	1	5.3	4.8
	A step forward	100	98.8	3	2.2	2.25	1	5.5	5
	Overall Avg.	**99.45**	**98.17**	**2.58**	**2.08**	**1.96**	**1.03**	**5.65**	**4.36**
	Std. Deviation	**1.35**	**0.6**	**1.53**	**0.39**	**0.66**	**0.08**	**0.45**	**0.51**

4.3 ($SD = 0.5$) for (1), 3.6 ($SD = 1.2$) for (2), and 3.8 ($SD = 1.4$) for (3). The study was approved by the University of Virginia–IRB (#4312).

3.4.4 Results

The participants designed a total of 27 usable gestures (9 *participants* × 3 *gestures*) using each method. For SequenceSense, participants were able to design all 27 gestures within the time limit, whereas for MAGIC, the participants could not complete the design of 4 gestures (two participants, two gestures each). Table 3.4 reports the average accuracy, the average false positive, the average number of iterations, and the average sequence length of the designed gesture using both the MAGIC and SequenceSense for the chosen gesture sets. It also reports the overall average in each category and their standard deviation. The complete list of original and modified gestures is in Table 3.5.

Quality of Designed Gestures

Figure 3.11 illustrates the gesture accuracy, false positive rate, and the gesture sequence length of the designed gestures using both methods. A repeated measures ANOVA revealed no statistical significance for gesture accuracy and false positive rate between the two methods.[1] This may indicate that both the methods are sufficiently comparable in recognizing gestures and minimizing false positives. However, the method effect was statistically significant on the designed gesture sequence length ($F(1,5) = 12.41,\ p < 0.05$). Using SequenceSense, participants were able to design usable gestures with shorter gesture sequences, implying more user-friendly gestures since shorter gestures are easier to remember and perform [25].

[1] p-value annotation legend in the reported figures (Fig. 3.11 - Fig. 3.12):: $*:\ 1.00e^{-02} < p <= 5.00e^{-02}$, $**:\ 1.00e^{-03} < p <= 1.00e^{-02}$, $***:\ 1.00e^{-04} < p <= 1.00e^{-03}$, $****:\ p <= 1.00e^{-04}$, and $ns : p >= 0.05$, not significant.

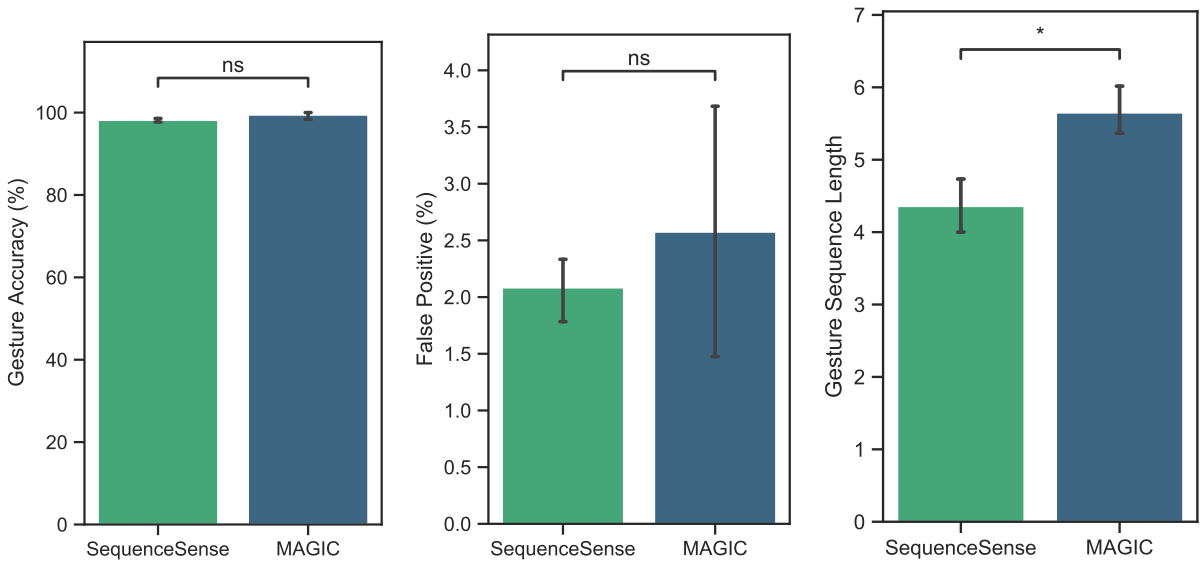


Figure 3.11: Quality of designed gestures. (a) Gesture Accuracy, (b) False Positive Rate, and (c) Gesture Sequence Length.

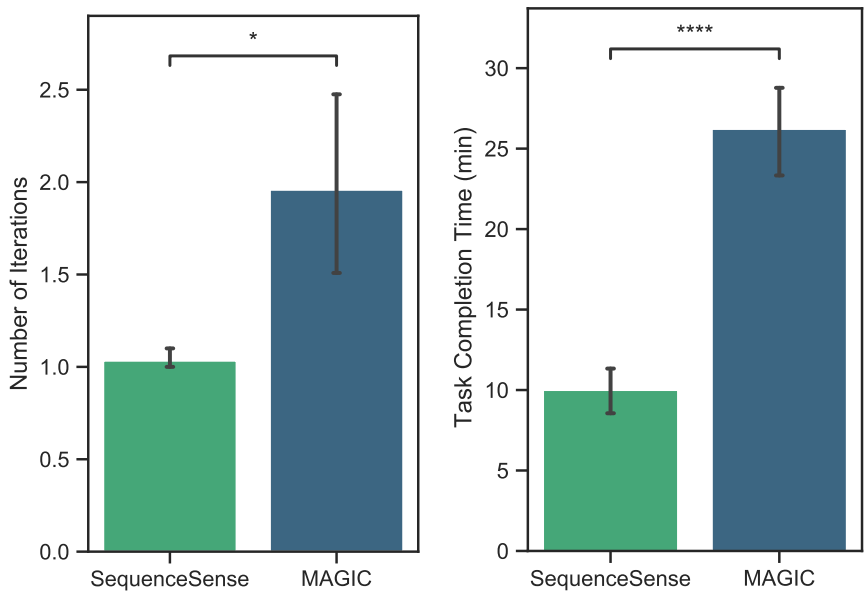
Efficiency of Designing Gestures

Figure 3.12 illustrates the number of iterations and the total time required for the participants to complete designing usable gestures using both methods. A repeated measures ANOVA revealed that the method effect was statistically significant for both measures. Participants required approximately twice the total number of iterations using MAGIC as compared to SequenceSense for designing usable gestures ($F(1,5) = 10.25,\ p < 0.05$). A larger effect was observed for task completion time, where participants took approximately 2.5 times longer to design usable gestures using MAGIC as compared to SequenceSense ($F(1,8) = 216.41,\ p < 0.00001$). These results hint at the efficiency benefits of using SequenceSense.

Overall User Experience

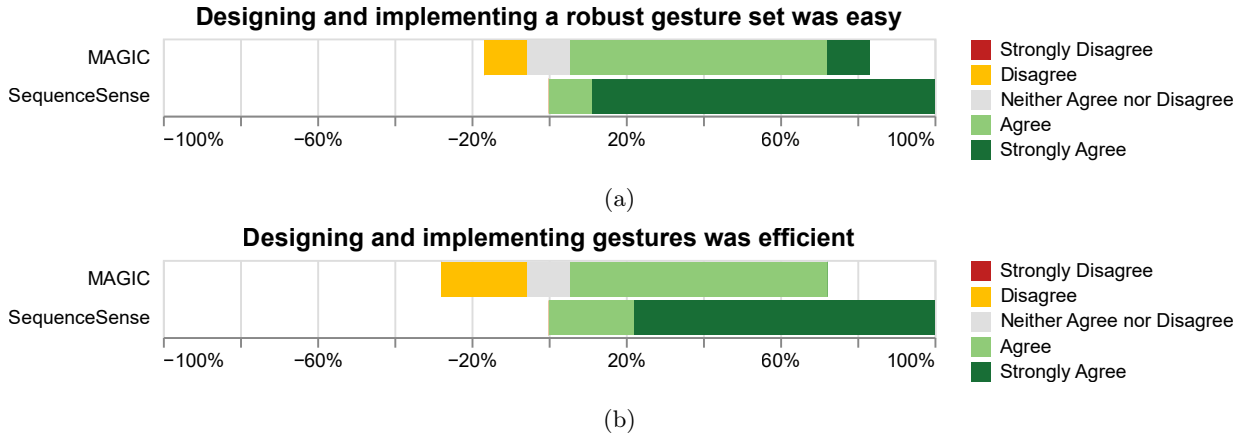
All participants in our study preferred using SequenceSense over MAGIC to design gestures. Through a 5-point Likert scale, participants reported their experience regarding ease of use and efficiency of designing and implementing gestures using both methods (illustrated in Figure 3.13). An Aligned Rank Transform (ART) [279] ANOVA revealed that the method effect was statistically significant for both ease of use ($F(1,16) = 23.21,\ p < 0.001$) and design efficiency ($F(1,16) = 28.31,\ p < 0.00001$) responses.

The participants appreciated the fact that they did not need to *"re-collect"* gesture samples multiple times when designing gestures using SequenceSense (P1, P2, P4, P5, P7, P9). *"I don't need to collect data very*



(a) (b)

Figure 3.12: Gesture design efficiency. (a) Number of Iterations and (b) Task Completion Time



(a)

(b)

Figure 3.13: Qualitative User Study Results. (a) Easy to Use and (b) Efficient

often; I can easily use the interface to merge many new gestures that I didn't even perform" (P2). Participants also appreciated the fact that SequenceSense required less effort (P1, P2) to design new gestures: *"there is less effort needed to come up with a good gesture, i.e., one that is easy to perform and is discernible"* (P2). Particularly, the participants commented on how they did not need to do *"trial and error"* (P4, P7) and *"guesswork"* (P9) when coming up with gesture modifications using SequenceSense's gesture editor as *"[SequenceSense] took the guesswork out of collecting data and having it fail"* (P9).

3.5 Discussion

All the designers rated SequenceSense positively. They found SequenceSense's gesture design process to be easier, more effective, and less time-consuming as compared to MAGIC, since SequenceSense did not require the designers to collect additional gesture samples for creating usable gestures. Since the designers can see the gestures as a sequence of atomic actions, they can easily understand which sequences conflict with daily activities and modify them with a simple change of atomic actions using the drag-and-drop gesture editor interface instead of recording an updated gesture. The sequence-based gesture recognizer is also efficient in storing the dataset and analyzing conflicts since it does not need the full sensor measurements.

No one would argue with the gesture design guideline that the gesture should be easy to understand, easy to use, having low false positive errors, and having low false negative errors [23]. However, designing such gestures can be challenging. Our study revealed that the user-friendly gestures [25, 9] may not be essentially usable gestures in real life, as similar movements may frequently happen during daily activities. However, I also saw a silver lining for which a small modification to such gestures could make them be significantly more usable; the simple addition of an atomic action to high-conflict gestures could significantly reduce potential conflicts. With SequenceSense, the iterative analysis and gesture modification could be done without additional user studies nor with performing and recording of the new gestures.

3.5.1 Design Implications

- **Easy-to-perform or intuitive gestures are not necessarily usable gestures.** SequenceSense utilizes the foot gestures from prior literature that were found to be instinctive and easy to perform [25, 261, 262, 263]. However, our analysis showed that these gestures can have many conflicts if used in everyday life (58.5% false activations), which can cause user frustrations. This highlights the importance of considering in situ gesture performance in the gesture design process.

- **Instant gesture performance analysis leads to a more optimized gesture design.** From the study results, I observed that the average length of the resulting gestures was shorter when designed using SequenceSense than MAGIC. I conjecture that this is due to the participants wanting to ensure the gestures are not triggered by daily activities without having to record the gesture samples again. With SequenceSense, the participants further optimized the gestures as they could immediately check the potential recognition performance of the gestures without recording gesture samples. Gesture design is an expensive process, and I believe that allowing gesture designers to modify gestures and inspect their performances will help the designers focus more on the usability of the gestures.

- **Testing is still essential to ensure usability.** While SequenceSense allows designers to quickly modify and inspect gestures to ensure that the gestures can be recognized at high accuracy, to design usable gestures, designers should still test the gestures by themselves or with other participants. In our study, I observed two participants who designed gestures with high-recognition performances finding the gestures difficult to actually perform. For example, a step backward followed by a quick foot tap could cause a loss of balance. After trying them, the participants modified the gestures. This shows the need for actually testing the gestures even with an advanced gesture-designing tool that provides an in-depth analysis of recognition performances. However, the gesture recognition performance analysis can help designers isolate the recognition performance and usability issues for a more efficient design process.

3.5.2 Limitations and Future Work

- I have demonstrated the use of SequenceSense for designing usable foot gestures; I believe that it can be used for other body-based gestures with some considerations. One of the biggest potential challenges of using SequenceSense for other body-based gestures is to segment movement into atomic actions. In SequenceSense, I used two types of behaviors as delimiters – 1) sudden directional or orientational changes in the movement (e.g., foot moving down after moving up) and 2) stable foot movement due to making contact with the ground. While the first behavior can be found in other body-based gestures, frequent contact with the ground or other stable surfaces is not very common for body movements other than those made with the foot, which may cause challenges in segmenting body movements. However, as similar behavior, which is raising and keeping the wrist still, has been used in smartwatch interfaces as a delimiter for wrist-based gestures [280, 281], I anticipate the method would also be useful in other body-based gesture interfaces.

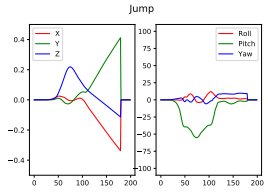
- I validated SequenceSense with 17-foot gestures chosen from the literature and further segmented them into atomic actions. These detected atomic actions do not guarantee that they are representative enough of other foot gestures not included in SequenceSense. Additionally, to develop SequenceSense, I have chosen foot gestures that are discrete in nature; however, a thorough evaluation and experiments are required to support more nuanced gestures.

- While the current SequenceSense implementation predicts the recognizer performance, such as accuracy, and false positive rates, it may be extended to predict the overall usability (e.g., the ease of use, fatigue). However, it would be challenging to incorporate the usability prediction feature into the system because the compound gesture is composed of atomic actions. The overall accuracy and usability of the compound gesture may be computed as the sum of the usability and accuracy of the individual

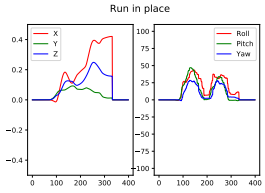
atomic action, which cannot be guaranteed since the difficulty of performing an atomic action may also depend on adjacent atomic actions. For example, *moving the foot to the right* may be easy when starting from the resting position, but it may be more difficult if the same atomic action comes after *foot forward* atomic action due to the position of the foot is farther away from the user's center of mass. However, if some models (e.g., estimating gesture execution difficulty by Vatavu et al. [282] and estimating shoulder fatigue for midair interactions by Hincapié-Ramos et al. [283]) take a sequence into account for predicting the usability/ease of use, then it may be possible to predict the overall usability of the compound gestures as well. It requires further in-depth investigation for ease of use in various aspects, which can be a possible future direction.

Table 3.5: The detailed similarity or dissimilarity metrics of the designed 27 gestures by the designers.

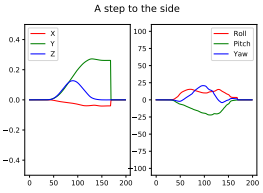
Participant	Method	Gestures	Accuracy (%)	False Positive (%)	Final Gesture Sequence	No. of Iterations	Duration (mins)	Is Complete?
P1	MAGIC	A step to the side	100	8	$a_0a_4a_0a_5a_0a_3$	2	25	Yes
		Turn left	100	0	$a_0a_{11}a_0a_{11}a_0a_4a_0a_{11}$	2		
		A step backward	100	3	$a_0a_3a_0a_{14}a_0a_7$	2		
	SS	Turn right	98	3	$a_0a_1a_0a_5$	1	8	Yes
		Rotate toes right	100	0	$a_0a_{11}a_0a_{10}$	1		
		A step forward	100	2	$a_0a_3a_0a_{14}a_0a_6$	1		
P2	MAGIC	A step to the side	100	0	$a_0a_3a_0a_{11}a_0a_{11}$	1	29	Yes
		Turn left	100	0	$a_0a_{11}a_0a_{11}a_0a_4a_0a_{11}a_0a_{11}$	3		
		A step backward	100	0	$a_0a_7a_0a_{11}a_0a_{11}a_0a_{11}$	2		
	SS	Turn right	98	3	$a_0a_5a_{w_4}$	1	10	Yes
		Rotate toes right	98	3	$a_0a_4a_0a_{10}$	1		
		A step forward	98	3	$a_{w_2}a_0a_6$	1		
P3	MAGIC	A step to the side	100	0	$a_0a_3a_0a_{11}a_0a_{11}$	1	30	Yes
		Turn left	100	1	$a_0a_{11}a_0a_4a_0a_{11}a_0a_{11}$	3		
		A step backward	100	0	$a_0a_7a_0a_{11}a_0a_{11}$	2		
	SS	Turn right	100	0	$a_0a_{11}a_0a_{11}a_0a_5$	2	13	Yes
		Rotate toes right	100	0	$a_0a_{10}a_0a_{11}a_0a_{11}$	1		
		A step forward	100	0	$a_0a_6a_0a_{11}a_0a_{11}$	1		
P4	MAGIC	A step to the side	100	6	$a_0a_3a_0a_{11}$	1	21	Yes
		Turn left	100	2	$a_0a_4a_{w_1}$	3		
		A step backward	100	0	$a_0a_7a_0a_{11}a_0a_{11}$	1		
	SS	Turn right	98	4	$a_0a_5a_0a_{12}a_0a_{13}$	1	10	Yes
		Rotate toes right	98	1	$a_0a_{10}a_0a_{12}a_0a_{13}$	1		
		A step forward	98	3	$a_0a_6a_0a_{12}a_0a_{13}$	1		
P5	MAGIC	A step to the side	–	–	—	–	30	No
		Turn left	100	3	$a_0a_{11}a_0a_4a_0a_{11}$	1		
		A step backward	–	–	—	–		
	SS	Turn right	98	2	$a_0a_4a_0a_5$	1	13	Yes
		Rotate toes right	98	3	$a_0a_4a_0a_{10}$	1		
		A step forward	98	3	$a_0a_6a_0a_7$	1		
P6	SS	A step to the side	98	2	$a_0a_{11}a_0a_3$	1	8	Yes
		Turn left	98	3	$a_0a_{11}a_0a_4$	1		
		A step backward	98	2	$a_0a_{11}a_0a_7$	1		
	MAGIC	Turn right	100	7	$a_0a_{11}a_0a_5a_0a_{11}$	1	25	Yes
		Rotate toes right	100	8	$a_0a_6a_0a_{10}a_0a_7$	4		
		A step forward	100	9	$a_0a_{10}a_0a_9a_0a_6$	2		
P7	SS	A step to the side	98	2	$a_0a_{11}a_0a_3$	1	9	Yes
		Turn left	98	1	$a_0a_4a_0a_{11}$	1		
		A step backward	98	2	$a_0a_{11}a_0a_7$	1		
	MAGIC	Turn right	–	–	—	–	30	No
		Rotate toes right	–	–	—	–		
		A step forward	100	0	$a_0a_6a_0a_{11}a_0a_{11}$	3		
P8	SS	A step to the side	98	4	$a_0a_3a_0a_{10}$	1	12	Yes
		Turn left	98	3	$a_0a_4a_0a_6$	1		
		A step backward	98	3	$a_0a_7a_0a_{11}$	1		
	MAGIC	Turn right	100	0	$a_0a_5a_0a_{11}a_0a_{11}$	2	28	Yes
		Rotate toes right	100	3	$a_0a_{11}a_0a_{10}a_0a_7$	3		
		A step forward	100	0	$a_0a_{11}a_0a_{11}a_0a_6$	3		
P9	SS	A step to the side	96	0	$a_0a_3a_0a_1$	1	7	Yes
		Turn left	98	3	$a_0a_4a_0a_6$	1		
		A step backward	96	1	$a_0a_7a_{w_1}$	1		
	MAGIC	Turn right	100	3	$a_0a_5a_0a_1a_0a_{11}$	1	18	Yes
		Rotate toes right	90	2	$a_0a_{10}a_0a_7$	2		
		A step forward	100	3	$a_0a_6a_0a_3$	1		



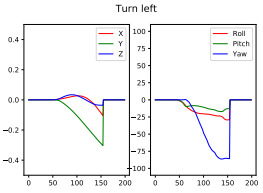
(a) Features of "Jump" gesture.



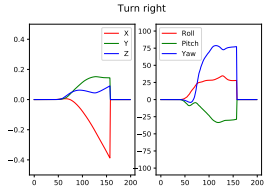
(b) Features of "Run in place" gesture.



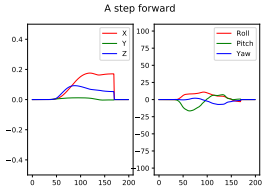
(c) Features of "A step to the side" gesture.



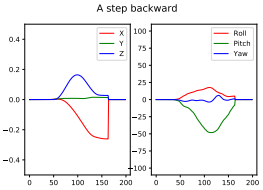
(d) Features of "Turn left" gesture.



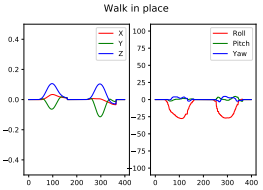
(e) Features of "Turn right" gesture.



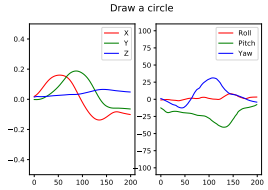
(f) Features of "A step forward" gesture.



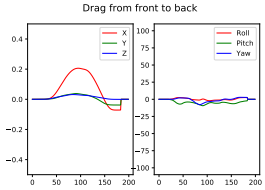
(g) Features of "A step backward" gesture.



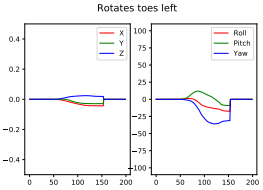
(h) Features of "Walk in place" gesture.



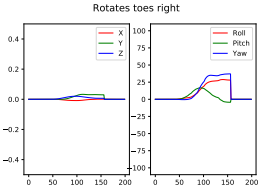
(i) Features of "Draw a circle" gesture.



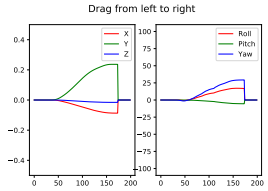
(j) Features of "Drag from front to back" gesture.



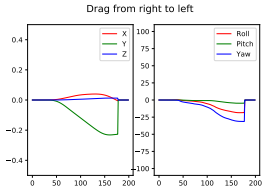
(k) Features of "Rotate toes left" gesture.



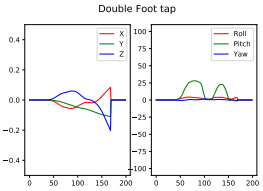
(l) Features of "Rotate toes right" gesture.



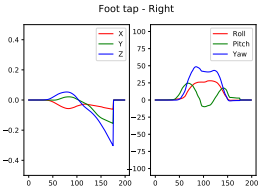
(m) Features of "Drag from left to right" gesture.



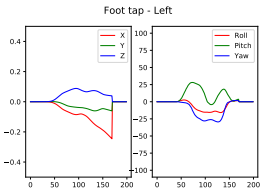
(n) Features of "Drag from right to left" gesture.



(o) Features of "Double foot tap" gesture.



(p) Features of "Foot tap to the Right" gesture.



(q) Features of "Foot tap to the Left" gesture.

Figure 3.14: Six-dimensional feature of the gesture. Distance measure in meters (left side) and orientation measure in degrees (right side) correspond to the number of samples for each of the subgraphs.

Chapter 4

Making Gestures Compatible Across Devices

The always-expanding proliferation of smart devices has allowed individuals to integrate technology into their ways of life in the form of wearables. Wearables like glasses, earphones, rings, watches, and pendants are now equipped with processors and sensors and can communicate with other smart devices. These integrations have enabled smart devices to facilitate quick microinteractions [4, 5, 6, 7, 8], more accessible interactions [9, 10, 11, 12, 13, 14, 15, 16], and even eyes-free interactions [17, 18, 19]. These interactions are often used across different devices for various reasons, including convenience, efficiency, and social acceptability. For example, Spotify allows users to control the music playing on smart speakers connected to a smartphone by using controls on their smartwatches [20], and researchers proposed using a wearable device as an input device for another wearable device [21, 22]. Modern wearables often act as an extension of our physical body as on-body interfaces to such an extent that people may build up muscle memory while performing gestures associated with those devices. However, these interactions are reliant upon the devices that support them, which could cause frustration when the user wears a non-smart device or forgets to wear the smart device. In such situations, the user has to find alternative ways of interaction, which may not be efficient. To make gesture interactions more usable by facilitating the users' reuse of muscle memory, which is what they gain through repetitive usage of gestures, the gesture should be made independent of specific devices. *To be specific, the gesture should be compatible across devices to facilitate the same outcome, even if the original gesture-sensing device is missing or not worn.*

This chapter proposes *UnifiedSense* to serve the purpose mentioned above. I propose utilizing the input gesture recognized on a wearable device (primary device) and the sensor data from all other connected

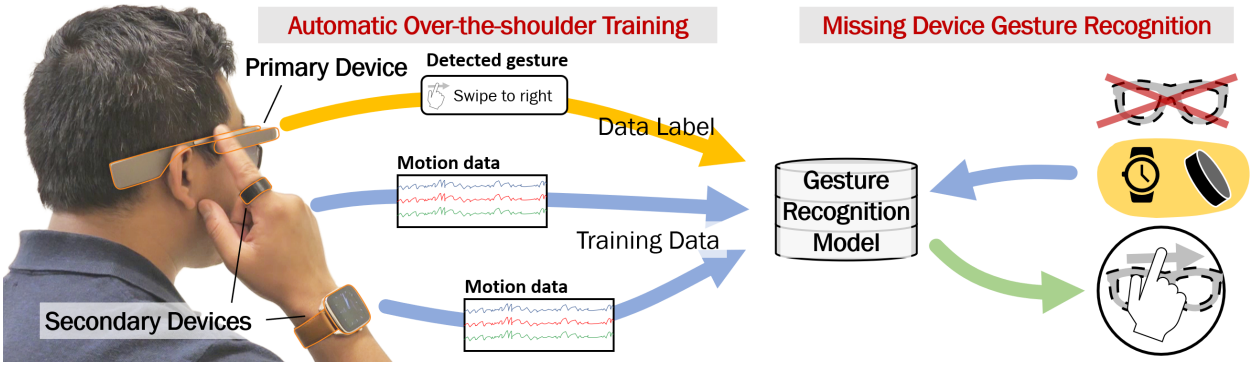


Figure 4.1: Over-the-shoulder training concept. When a user uses gestures on a smart device (primary device), data from other devices (secondary devices) will also be collected to train a unified gesture recognizer model. The gesture recognized by the primary device will be used as a label for training. Once the gesture recognizer is sufficiently trained, the users can use primary device gestures without wearing it.

wearable devices (secondary devices) to train a unified gesture recognition model, *UnifiedSense*, to enable interactions for the primary device even when the device is absent. As wearable devices are becoming more prevalent, it is not uncommon to see people wear multiple wearable devices. These wearable devices are equipped with various sensors such as inertial measurement units (IMU), microphones, and touch sensors. While worn on different locations on the body, their sensing is not completely exclusive and is often relatable to each other. For example, tapping the headphone's touchpad involves the movement of the hand, which can be measured by the smartwatch's IMU on the wrist, and the collision between the finger and the headphone, which can be measured by the IMUs on the smartwatch, smart glasses, and the headphones. As many smart wearable devices connect to a central device, such as a smartphone, I believe that the data from those sensors can be collected together along with the label created by the intended interaction device to train a unified gesture recognition model (Figure 4.1). I named this process *over-the-shoulder training*.

The idea of exploiting relatable actions is not new; researchers have demonstrated the feasibility of indirectly sensing gestures using devices on a user [10, 37, 284]. For example, Scott et al. used a mobile device in the pocket to detect foot gestures [10], and Xu et al. used earbuds to detect touch gestures on face [37], without attaching a sensor or a device at the location of interaction. While promising, these methods only work for a specific configuration and require a user to train the model before using the gestures. However, given the diversity of the types of devices one can wear today and the variety of configurations available for a user, it may not be practical for a user to train the model each time they have a new device or wear a different set of devices on them. Our over-the-shoulder training, on the other hand, lets the system train the model by itself by collecting training samples as users perform gestures on wearable devices. I assume that a wearable device can reliably recognize input gestures designed for it (e.g., touch gestures on a smartwatch) and the recognition result can be used for labeling a gesture training sample. As the user continues using

gestures on their wearable devices, the unified gesture recognizer will be trained automatically over time.

Once the model is sufficiently trained, users can use gestures associated with a primary device without wearing it. To verify the feasibility of our method, I implemented a gesture recognition system that collects data from multiple body-worn devices. When the gesture recognition system detects a gesture on one of the primary devices (e.g., touchpad on smart glasses), it segments movement data from the IMUs on other devices. I used segmented data to train a Time-Series Transformer (TST) [2] model to classify gestures. For validation, I conducted an experiment with 15 participants. Our system collected sensor data from four wearable devices (headphones, smartwatch, smart ring, and smart glasses) with the participants performing five touch gestures (tap and four directional swipes) on the devices. The experiment results showed that, if a user performs gestures for the first time, the system can only detect gestures at an average accuracy of 81.01% using a model trained by other users wearing the same configuration of devices. However, if the new user continues to use the gestures, after the gesture set was used 20 times by itself, the recognition accuracy increases to 95.2%.

Designing a practical unified gesture recognition model that can be used *in the wild* comes with the inherent challenge of eliminating false gesture activations associated with natural body movements. To address this, I implemented a binary classifier using the TST model to differentiate intended gestures and natural body movements. To train the binary classifier, I collected 20 hours of regular activity data from 12 users. When tested with artificially generated real-time data that include both natural movements and intentional gestures, the system could recognize the gestures at an average accuracy of 90.9% with a gesture recognizer trained with 22 samples per gesture.

UnifiedSense uses the data collected from secondary devices while the users perform gestures on primary devices. Therefore, if the user adds a new device or changes the device configuration, the model may not reliably recognize the gestures until enough interactions are performed on the primary device while the user wears the new setup. While the training can be done within a few days (see Section 4.4.1), this process can be confusing to users. Therefore, I developed an Android app that shows the readiness of the model (i.e., the likelihood of properly recognizing gestures) so that the user can know if they can start using the gestures without the primary device.

4.1 UnifiedSense

UnifiedSense uses data from heterogeneous sensors and the true label obtained from the primary device to train the model, which I call an over-the-shoulder training approach. For instance, consider performing a touch gesture on a smartwatch. When a user moves their hand to perform the gesture, the sensor data from

other on-body smart devices, such as a smart ring or a smart glass, can be leveraged to learn the hand's movement pattern. For this interaction, I consider the smartwatch as the primary device, whereas all the other devices are considered secondary devices. Similarly, when a gesture is performed on another on-body device, that device becomes the primary device, whereas all the other devices are considered secondary devices. Such scenarios can potentially generate more alternative ways to learn gestures performed for the primary device by other available sensors. I can merge those alternate ways to build a unified gesture recognizer through our method.

4.1.1 System Design

UnifiedSense system has two main components, one for training the personalized gesture recognizer model and another for recognizing gestures. For model training, the system collects sensor data from the secondary devices and uses the primary device's input as a label for the collected data to be used for the training sample. Once the model is trained on enough data, UnifiedSense uses the trained model to recognize gestures for the primary device.

UnifiedSense recognizes gestures in two steps. First, a gestures detector judges whether a gesture is present. If a gesture is detected, it is then recognized by a classifier. Figure 4.2 illustrates the overall pipeline of the system.

Device Implementation

Our implementation assumes a scenario in which a user has four wearable devices: a smartwatch, smart glasses, smart headphones, and a smart ring, although the method is not limited to the assumed scenario. For the proof-of-concept implementation, I used a set of custom sensor devices that mimic a smartwatch, a smart ring, and smart headphones for efficient real-time multi-sensor data acquisition. A smartwatch was implemented by attaching a 9-DOF IMU (Sparkfun BNO080) and a touchpad (28×33 mm) to a Velcro band (Figure 4.3a). An Arduino Uno microcontroller was used to collect the measurements and send them to a laptop. A smart ring was implemented by attaching a 9-DOF IMU (Sparkfun BNO080) to a Velcro band (Figure 4.3b). To simulate the smart headphones with touch input, I attached a touchpad (46×64 mm) on one side of the headphones (Figure 4.3c). *Google Glass XE* (Figure 4.3d) that has a touchpad on its frame was used as smart glasses. The four devices were connected to a laptop computer for data collection and training of the unified model. Arduino Uno microcontrollers and the touchpads were connected to the laptop using USB cables. The smart glasses communicated to the laptop through Wi-Fi using UDP protocol. The complete system setup is shown as a block diagram in Figure 4.4.

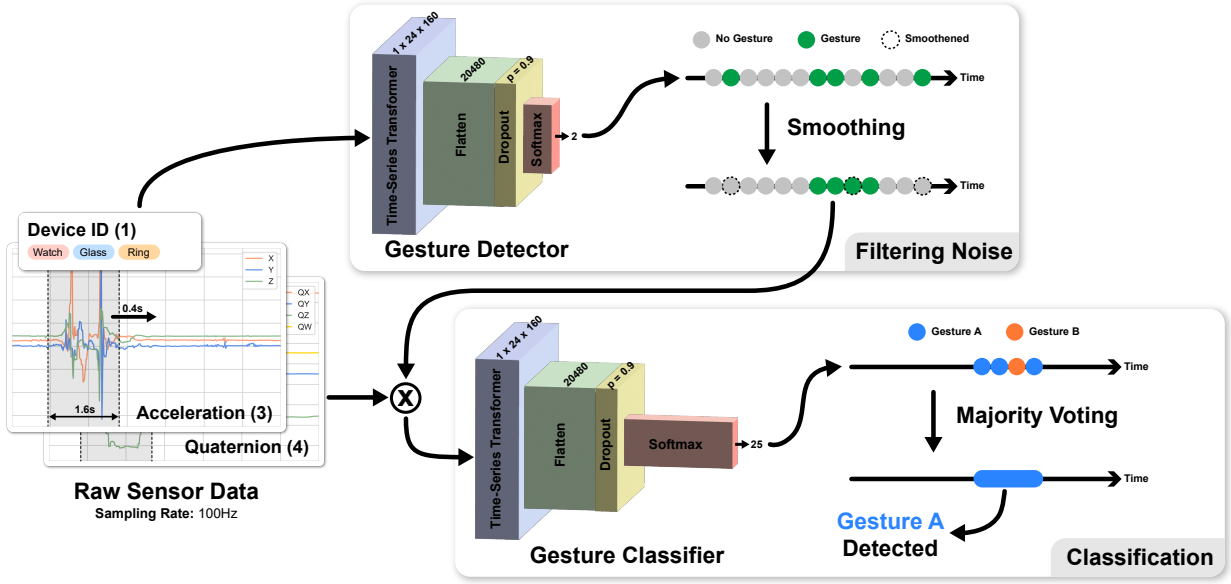


Figure 4.2: UnifiedSense pipeline overview. A sliding window with a duration of 1.6 s and a step size of 0.4 s is applied to the sensor data. Raw features are extracted, combined, and fed into a binary TST classifier (Gesture Detector). A majority voting scheme is used to merge adjacent sequences of consecutive 1's (green) and handle noise (Smoothing). Gestures are identified when there are at least 3 consecutive 1's (Filtering Noise). Detected gestures are further classified (Gesture Classifier), and another round of majority voting (Majority Voting) is employed for final gesture recognition.

Figure 4.3: Simulated smart devices: (a) smartwatch – implemented with an IMU, Velcro band, and a touchpad, worn on the wrist, (b) smart ring – implemented with an IMU and Velcro band, worn on the index finger, (c) smart headphones – attached with a touchpad on them, and (d) Google Glass XE.

Device Configuration

In reality, people may wear different sets of wearable devices. To test the feasibility of our method used in different configurations, I set up five configurations with different primary (P) and secondary (S) devices. The considered five configurations are: *Configuration* 1 *(C1)* – P: smart glasses, S: smart ring and smartwatch. *Configuration* 2 *(C2)* – P: smart headphones, S: smart ring, smartwatch and smart glasses. *Configuration* 3 *(C3)* – P: smartwatch, S: smart ring and smart glasses. *Configuration* 4 *(C4)* – P: smart headphones, S: smartwatch, smart ring and smart glasses. *Configuration* 5 *(C5)* – P: smart glasses, S: smartwatch and smart ring.

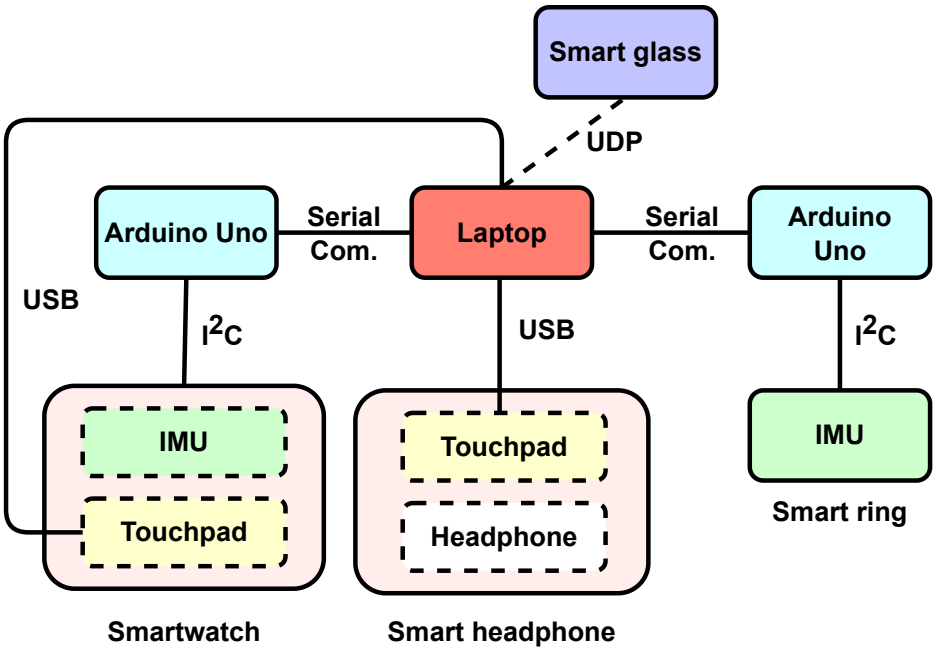
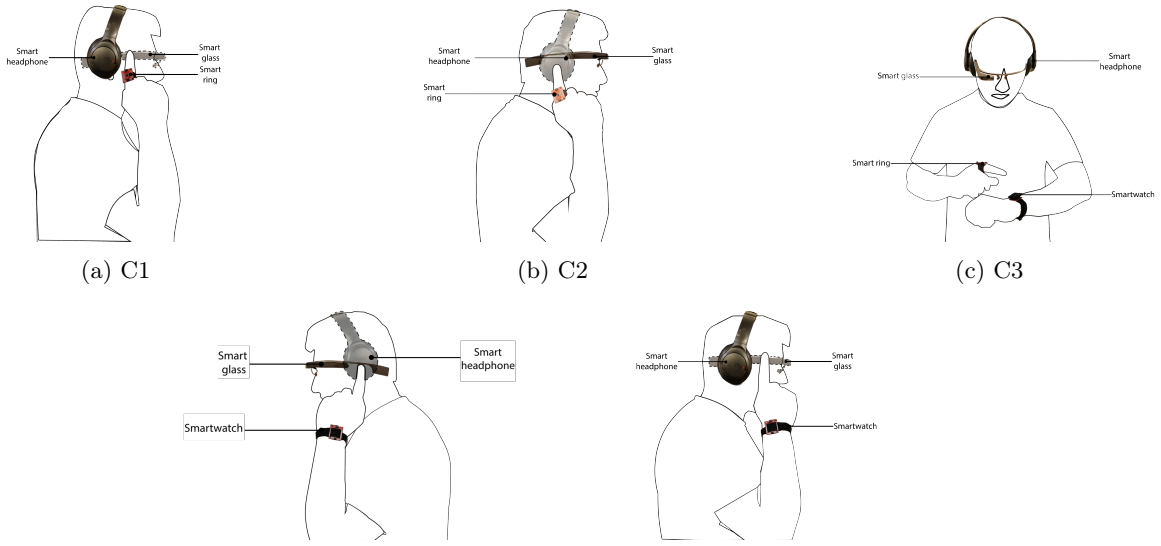


Figure 4.4: The block diagram of the UnifiedSense system showing sensor connections and communication protocols.

The smart ring was worn on the user's right index finger, and the smartwatch was worn on the user's left wrist (C1 - C4). For C5, the primary device was the same as C1, and the secondary devices were worn on different hands (smartwatch on the right wrist and smart ring on the left index finger). Please refer to Figure 4.5 for the configurations in more detail. Prior work by Gu et al. [285] has shown that an IMU-based ring worn on the index finger can accurately sense touch contact. Based on this, I placed the smart ring on the index finger in our experimental setup. Note that participants in the experiment wore all devices for all configurations, but the data was selectively collected based on the configuration.

Gesture Detection

Gesture detection starts by setting a 1600-ms sliding window with a step size of 400 ms. Twenty-four raw features are extracted and combined from the window and fed into a binary TST classifier. The classifier outputs 1 if the sensor content belongs to a gesture and 0 otherwise. To reduce noise, UnifiedSense uses a majority voting scheme, where adjacent sequences of consecutive 1's are merged if separated by one or two 0's. A gesture is present when there are 3 or more consecutive 1's, corresponding to a minimum gesture duration of 1200 ms. Whenever a gesture occurs, UnifiedSense feeds those 1600-ms segments into the gesture classifier again and uses another majority voting scheme for the final classification of the gesture. The first occurring gesture is used as a tie-breaker during the final majority voting scheme.



(a) C1 (b) C2 (c) C3

(d) C4 (e) C5

Figure 4.5: Different configuration setups for collecting sensors data while performing the touch gestures to the primary device.

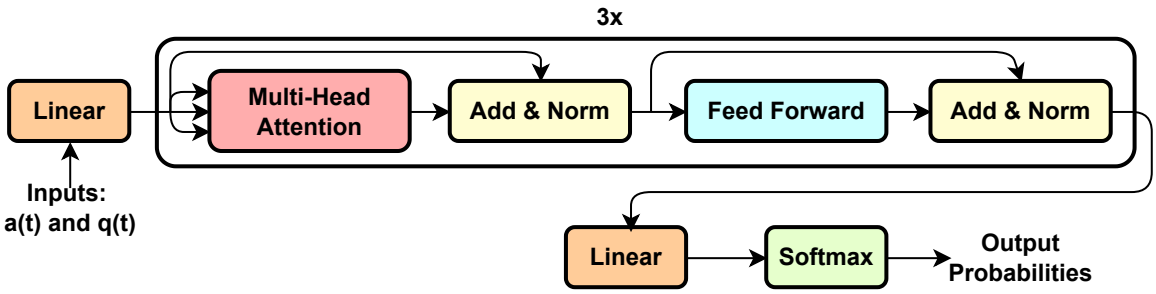


Figure 4.6: TST [2] model architecture based on Multi-Head Attention [3].

Classifier

I adapted Time-Series Transformers (TST) for gesture classification, as recent research has shown it to outperform other deep learning classification models for multivariate time-series classification [2]. The basic architecture of the TST model follows the original transformer work by Vaswani et al. [3]. The core architecture of TST depends on a transformer encoder. However, TST does not use the decoder part of the architecture. A schematic diagram of the generic part of the TST model (with adaptation) is illustrated in Figure 4.6. Here, I briefly describe how I adapted the TST model for our purpose.

First, the input feature vectors x_t are normalized (for each dimension, the normalization is done by subtracting the mean and dividing by the variance across the training set samples) and then linearly projected onto a d-dimensional vector space, where d is the dimension of the transformer model sequence element representations (typically called model dimension). Second, outputs from this layer become the queries,

keys, and values of the self-attention layer after adding the positional encoding and multiplying by the corresponding matrices [3]. Third, since the transformer is a feed-forward architecture that is insensitive to the ordering of input, to make it aware of the sequential nature of the time series, TST adds positional encodings [2]. Fourth, the next layer performs batch normalization. Finally, outputs from the previous layer are fed to a linear layer and then perform a softmax for classification.

In our implementation of TST, I used 3 parallel attention heads (Figure 4.6). To alleviate the over-fitting issue during training, I used two separate dropouts [286] in the model. A residual dropout ($p = 0.4$) was applied in the encoder (Multi-Head Attention). Another dropout was applied to the classification block's final fully connected ($p = 0.9$) layer. Our TST model was trained for 20 epochs on the training set. I use a batch size of 64, with a categorical cross-entropy loss function and the Adam optimizer. Both the dropout parameters (residual dropout, $p = 0.4$, and fully-connected dropout, $p = 0.9$) and batch size (64) were empirically decided. I used *GELU* [287] activation function throughout our implementation. In total, there are 425,730 parameters in the model.

Real-Time System Implementation

UnifiedSense was tested on a laptop with an Intel Core-i7 processor (2.60 GHz, RAM: 16 GB) to perform real-time gesture recognition. The smart glasses send acceleration and orientation data through UDP in 10-ms chunks, while the smartwatch and smart ring send data through a serial port in 10-ms chunks. The three channels' data are combined and fed into the gesture detection and classification pipeline, and zero padding is applied for absent channels. The pipeline runs on the CPU and has an average computation time of 30 ms, with an average delay of 600 ms between gesture completion and classification result.

4.1.2 Data Preprocessing

Data Labeling

UnifiedSense collected the timestamp and the detected gesture from the primary device to segment and label the time-series data (3-axis acceleration, $a(t)$, and orientation in quaternions, $q(t)$) from secondary devices. Our system used a segmentation window with a duration of 1.6 s for each gesture, which includes 100 samples before the trigger and 60 samples after the trigger (Figure 4.7). The segmentation window size was chosen empirically based on the gesture-classifying accuracy tested during the development (Section 4.1.2).

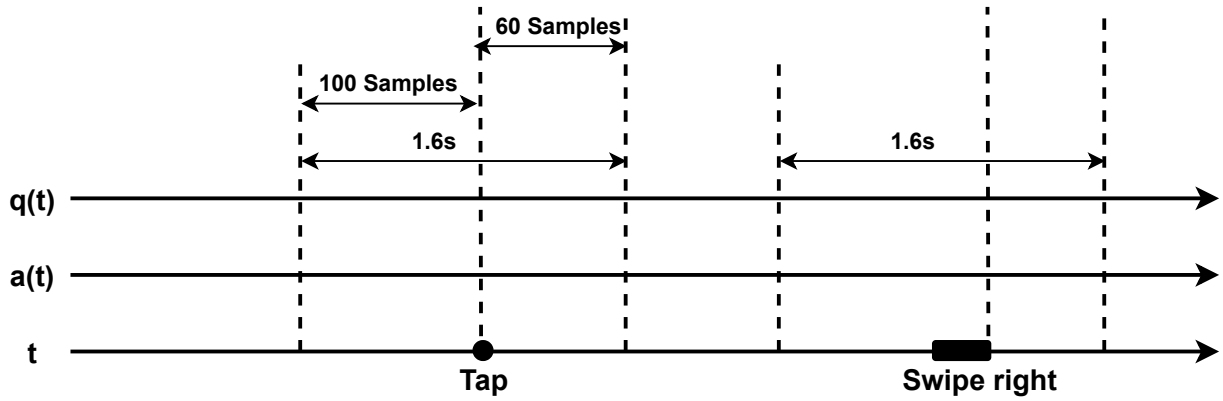


Figure 4.7: Data labeling process: UnifiedSense segments and labels the data from secondary devices using the gesture input detected on the primary device.

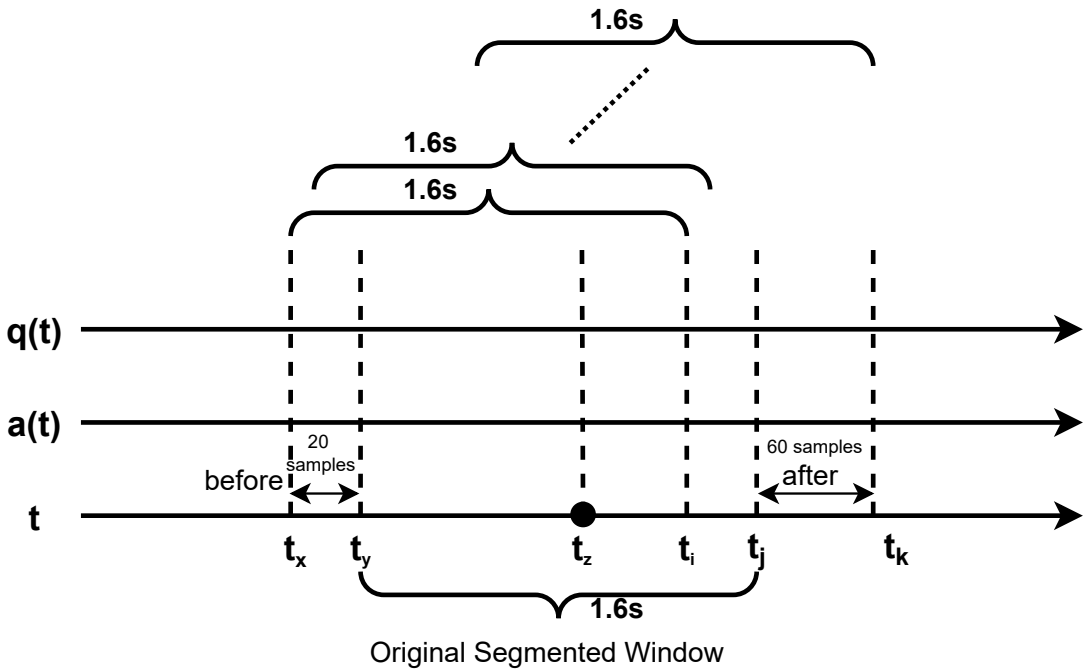


Figure 4.8: Data augmentation process: UnifiedSense augments data by cropping overlapping windows.

Data Augmentation

After data segmentation, I used several data augmentation techniques to generate sufficient samples. First, I generated overlapping samples from the original segmented window sample (from time t_y to t_j, where the gesture was triggered at time t_z) by increasing the segmentation window by 20 samples before the original segmented window and 60 samples after the original segmented window (Figure 4.8). This uneven enlargement was made to keep the original segmented window in the middle (from time t_y to t_j) by ensuring the trigger from the primary device in the center. Keeping the original segmented window as a center, I increased 80

samples from one sample, starting from time t_x with a sliding window size 1.6 s with hop size 1 sample. For example, the first sample was cropped from time t_x to t_i, the second sample was cropped from time t_{x+1} to t_{i+1}, and so on. Second, I used two time-series data augmentation techniques [288] such as time-warping to simulate gesture temporal variance, with 2 interpolation knots, and warping randomness was used to increase the number of samples by two times. I also employed the same augmentation techniques to increase the number of negative samples.

Uniform Input Shape

UnifiedSense uses a device identifier (id), 3-axis accelerations ($a(t)$), and device orientation presented in quaternion ($q(t)$) from each device to be fed into the TST model. The data preprocessing pipeline ensures that the input to the TST model is consistent through all five configurations. To make the consistent input shapes for all the gestures across the configurations, I combined sensor data from three devices (smart ring, smartwatch, and smart glasses) into 24 tuples (8 tuples from each device). The preprocessing phase applies zero padding for devices not present to maintain consistency in input shape.

Window Size Calculation

I determined the window size for our gestural data by conducting an empirical analysis and using a 70-30 train-test split. I used two parameters, pre-length and post-length, with a range of 0-120 to fix the window length. I generated all combinations for these parameters and trained and tested the classifier for each window size to determine the highest accuracy. Our analysis found that the highest accuracy was achieved by several combinations, such as [120, 100], [100, 60], and [100, 100]. I ultimately chose to use a pre-length of 100 samples and a post-length of 60 samples as the window size to reduce computational time and allow for real-time recognition. The time-series input shape for our model is (24, 160), with a window size of 160, and a dimension of three input sensors is 24 (3×8).

4.1.3 System Implementation

UnifiedSense consists of four major components: data collection, segmentation, training unified gesture recognizer, and real-time gesture detection and recognition. All the components were implemented using *python*. The data collection and segmentation components were merged into the same pipeline. A program with three separate threads was implemented to collect sensor data from three IMU-based devices, such as a smartwatch, smart ring, and smart glasses. In the same program, another three threads were implemented for collecting the touch input data from the touchpads (one for headphones and another for smartwatches) and

smart glasses. All six threads ran simultaneously and received the sensor and touch input data from all the devices. When our system detected any touch and the corresponding type from a primary device, it segmented the data using the system time by cropping the 100 samples before and 60 samples after the touch input detection and then labeled the data accordingly. Later the labeled data was saved into files with different threads. It is worth mentioning that UnifiedSense also stored all the raw data received from the IMU-based sensors and the touch input type from the primary devices. It allowed us to analyze the data to determine the best window size for building the unified gesture recognizer. I analyzed the data and built a unified gesture recognizer offline using *PyTorch* [289] python library. I also built a binary model to incorporate it in the pipeline (Figure 4.2) for real-time gesture detection and recognition. Once the binary and gesture recognition models were built, I used those pre-trained models to make a pipeline for real-time gesture detection and recognition implementation. This pipeline was also implemented with Python and PyTorch-library. An Android application was implemented for gesture detection on the Smart Glass by leveraging the legacy of Google Glass's touch detection algorithm. This application also collected accelerations and orientations in quaternion. Finally, the sensor data (IMU and touch input) were sent to the laptop using the UDP protocol.

4.2 Evaluation

I conducted three data collection studies to develop and evaluate UnifiedSense. The first study focused on collecting data from users performing gestures while wearing multiple wearable devices to build a gesture recognition model and test its gesture classification performances. The second study was conducted to collect negative data from users performing regular activities to make the model distinguish gestures and false activations and to test the performance of the model in a more realistic setting. The third study was conducted to understand the recognition performance characteristics when users use UnifiedSense while wearing a device similar to the primary device and not wearing the primary or similar device. In this section, I describe the details of these studies and discuss the evaluation results.

4.2.1 Gesture Data Collection

To build a gesture recognizer model for UnifiedSense, I conducted a user study to collect sensor data. I chose five basic touch gestures for data collection and evaluation. These are tap and four directional swipes: up, down, left, and right. The data collection study was held in a single session. The study was approved by the institutional review board (IRB) and fully complies with national laws and institutional regulations related to COVID-19.

The goal of the study was to collect the sensor data from the smart wearable devices while performing the touch gestures for the primary devices. Our system collected different sets of sensor data for each configuration from the connected devices at a sampling rate of 100 *Hz*. These devices collected 3-axis acceleration (raw acceleration) and its orientation in quaternion calculated by the AHRS (Attitude and Heading Reference System) algorithm implemented in the BNO080 module and Android sensor framework. The collected orientation output has no specific reference for heading, while roll and pitch are referenced against gravity, as it is produced by fusing the outputs of the accelerometer and the gyroscope (i.e., no magnetometer). Besides, our system collected touch gesture types and triggered timestamps to label the sensor data. As the time-series sensor data tuples are the same across four devices, our system recorded the samples from different devices with a device identifier (id) tag. The device identifier tags for the smartwatch, smart ring, smart glass, and smart headphones were 1, 2, 3, and 4, respectively. The collected movements and gesture types for each configuration are reported in Table 4.1. The collected size of the times-series data tuple is 16, 24, 16, 24, and 16 for the configurations C1-C5, respectively, excluding the timestamp. The dataset is publicly available.[1]

Table 4.1: Collected sensor data and gesture type for each configuration from each smart device. t refers to the timestamp, $a(t)$ refers to the 3-axis acceleration (3-tuples), $q(t)$ refers to the orientation in quaternion (4-tuples), and gt refers to the gesture type (e.g., tap, swipes).

	Devices			
Configuration	**smart glasses**	**smart ring**	**smartwatch**	**smart headphones**
C1	<t, id, gt>	<t, id, a(t), q(t)>	<t, id, a(t), q(t)>	–
C2	<t, id, a(t), q(t)>	<t, id, a(t), q(t)>	<t, id, a(t), q(t)>	<t, id, gt>
C3	<t, id, a(t), q(t)>	<t, id, a(t), q(t)>	<t, id, gt>	–
C4	<t, id, a(t), q(t)>	<t, id, a(t), q(t)>	<t, id, a(t), q(t)>	<t, id, gt>
C5	<t, id, gt>	<t, id, a(t), q(t)>	<t, id, a(t), q(t)>	–

Participants

A total of 15 participants ($Male = 10$, $Female = 5$) ranging from 23 to 34 years of age ($Mean = 29.76$, $SD = 4.3$) were recruited from a local university. All participants were right-handed. Participants were given $25 each for their participation.

Procedure

Researchers helped participants to wear all devices with Velcro bands properly, helping them to adjust the wearing until they felt comfortable with the devices. After the participants wore the devices, a researcher demonstrated how to perform the five gestures for each configuration to the participants. Then the participants

[1] https://github.com/aashikazim/UnifiedSense_DataSet

practiced using the gestures for 5 minutes to familiarize themselves with the gestures for the five configurations. Then, participants were asked to perform the five touch gestures on the primary device for each configuration. Participants were asked to perform each gesture 40 times in each configuration. The participants performed the gestures from the resting position, and they were asked to move their hands back to the resting position after performing each gesture. Between each gesture, the participants were asked to rest for 2 seconds to avoid consecutive repetitions of the same movement, which may lead to the overfitting of the model. Once all gestures were performed in one configuration, the participants were given a 2-minute break before performing the gestures for the next configuration. During the break, researchers made necessary changes in wearing the devices in different places if required. After the break, the participants performed the gestures on a primary device of the new configuration. As a result, I collected a total of $15,000$ gesture samples (5 *configurations* $\times$ 15 *participants* $\times$ 5 *touch gestures* $\times$ 40 *samples*). The study lasted about 75 minutes on average for each participant.

This study served as the foundational step in developing a robust gesture recognition model. By collecting data on various gestures performed by participants, I obtained a dataset that allowed us to train and fine-tune the gesture recognition algorithms. This study laid the groundwork for subsequent studies by providing the necessary analysis and model development dataset.

4.2.2 Negative Data Collection

I conducted another study to collect negative data to prevent the gesture model from causing false gesture activations during daily activities. I recruited 12 participants ($Male = 10$, $Female = 2$) from a local university where 11 of them were participants of the previous study. All participants are right-handed and their age range is between 24 to 35 ($Mean = 30.03$, $SD = 4.9$). Researchers helped the participants wear the devices. I used a 4-meter-long USB extender to connect wired devices to the laptop. Therefore, the participants performed their daily activities within the perimeter of a 4-meter circle, which is similar to the size of the living room where the study was conducted. In this session, I asked participants to perform normal indoor daily activities in sitting and standing conditions, such as walking, phone browsing, typing, or working on a laptop. They were also asked to behave freely for about an hour and a half. I observed that the participants performed activities such as, leaning on the chair, exercising, moving a rolling chair, nudging their head, scratching head/hands, using a mouse/touchpad, playing/writing with pens, opening bottles, using phones, and drinking water. The total negative samples amounted to about 20 hours. The participants received $15 for their time.

4.2.3 Offline Analysis

I first performed an offline analysis to understand the performance of the gesture recognizer.

Population (within-subjects) Results

In our study, I opted for a personalized model in the offline analysis for several reasons. Firstly, our intention was to mimic the online training process where each participant's data is used to train the model individually. By employing a personalized approach, I aimed to assess the model's performance in detecting gestures for individual users and evaluate its robustness in handling missing devices. This decision allows us to understand the effectiveness of personalized training in capturing user-specific gesture patterns and adapting to their unique device configurations. Furthermore, it provides insights into the potential of the UnifiedSense system to accommodate diverse user preferences and device combinations, enhancing its applicability and usability in real-world scenarios.

To make a personalized model, I combined data in a leave-one-participant-out (LOPO) fashion with a slight moderation, i.e., first, I combined data for 14 participants. Next, from the leave-one-participant (LOP), I randomly selected 50% of data (50-50 train-test split). Then, I combined these 50% data from the LOP with other 14 participants' data to make a final training set. Finally, I used the remaining 50% data from the LOP as a test for the gesture classification. To investigate the personalized models, I trained and tested our gesture classification model (TST) 15 times with different participants' setups. Using these train-test splits, our TST model achieved 95.2% ($SD = 2\%$) of accuracy and 94.7% ($SD = 1.99\%$) of F1-score, on average.

Figure 4.9 shows the confusion matrix for the 5 gestures using moderated LOPO train-test split as mentioned earlier. It is interesting to report from the confusion matrix that there are almost no inter-configuration conflicts between gestures. I used a device identifier for each device, which allows the classifier to distinguish between configurations. However, a few number of intra-scenario gestural conflicts exist, which are less than 2%.

In addition, to estimate the general effectiveness of our TST model in classifying gestures, I performed cross-validation [290, 291]. Specifically, I performed a stratified 10-fold cross-validation with an 80-20 train-test split by mixing all the 15 participants' gestural data. In this case, the gesture classification model (TST) achieved an average of 96.1% ($SD = 2.2\%$) accuracy and an average F1-score of 95.1% ($SD = 1.8\%$). Though I measured the model's effectiveness with a 10-fold cross-validation, the results found with the personalized model (discussed above) are used for further analysis in later experiments.

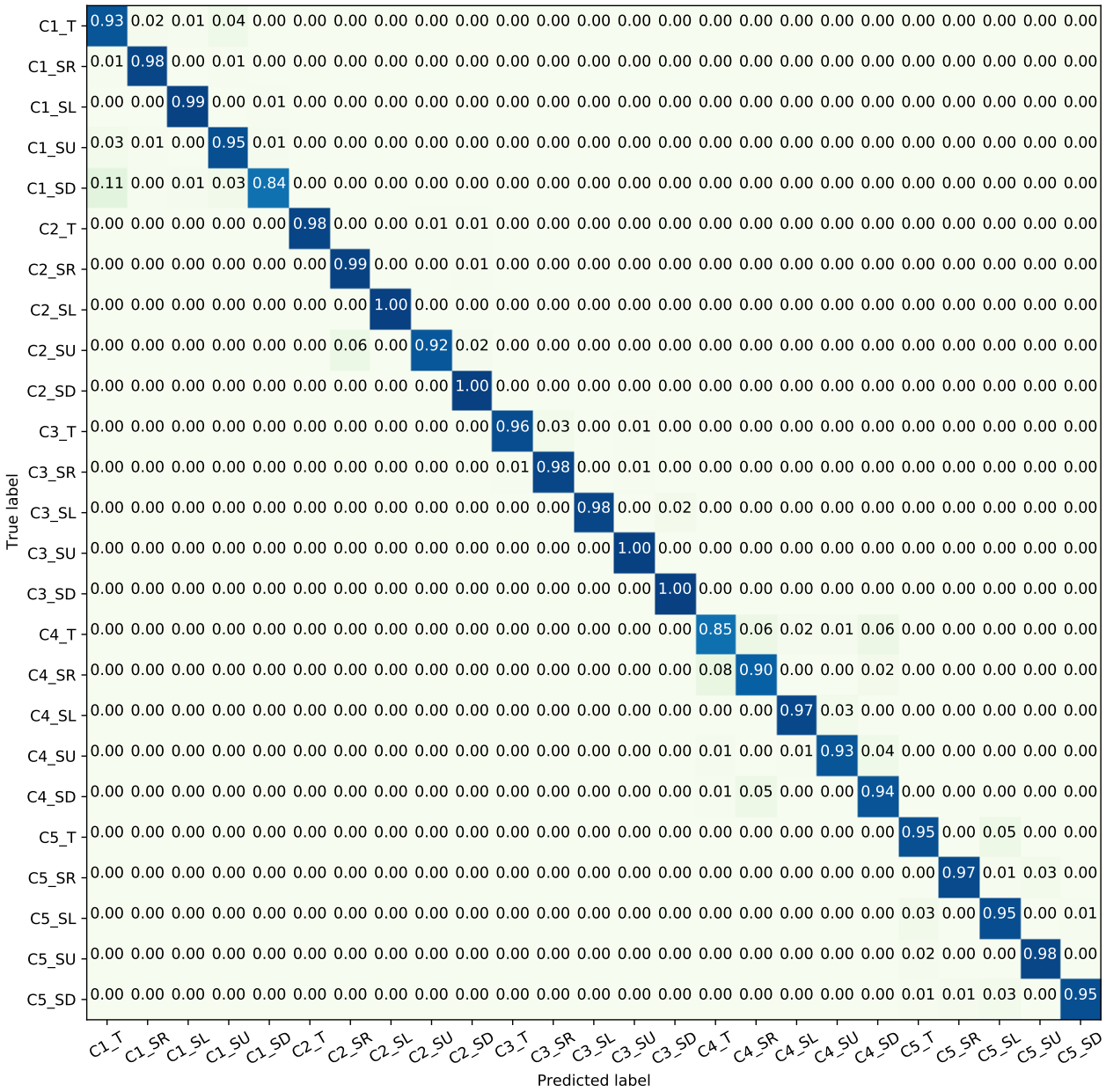


Figure 4.9: Confusion Matrix (CM). The labels are annotated as T for Tap, SR for Swipe Right, SL for Swipe Left, SU for Swipe Up, and SW for Swipe Down. Here, C1 - C5 represent the configurations. For example, C1_T represents the tap gesture performed in C1, C5_SR represents the swipe right gesture performed for the C5, and so on.

Leave-One-Participant-Out (between-subjects) Results

The time-series data of acceleration and orientation for the same gesture can appear different across users for a couple of reasons: (a) users may perform gestures in unique ways, or (b) users' unique body structure can produce different orientations while performing the gestures, or (c) users' gesture performing speed may vary from user to user. To investigate the feasibility of our model that could recognize gestures by new users, I

trained our model using leave-one-participant-out (LOPO) cross-validation. This gave us an overall accuracy of 81.01% ($SD = 1.2\%$) (F1-score: 80.02% ($SD = 1.4\%$)), a 14.19% drop in accuracy from the best-trained model within-subjects.

UnifiedSense will automatically train the model as the user performs gestures. To understand how the new user's gesture use improves the gesture recognition performance, I conducted an analysis by saving our leave-one-participant-out model and training it on a few examples of each gesture from the ignored participant. Figure 4.10 illustrates average accuracy and F1-scores by the number of additional training samples. It is evident from the figure that, with just eight gestures, the performance accuracy and F1-score improved from 81.01% to 90.3% and 80.02% to 90%, respectively. The performance approached near the population test accuracy with additional samples, reaching 94.2% accuracy and 94.1% F1-score with 18 additional gesture samples.

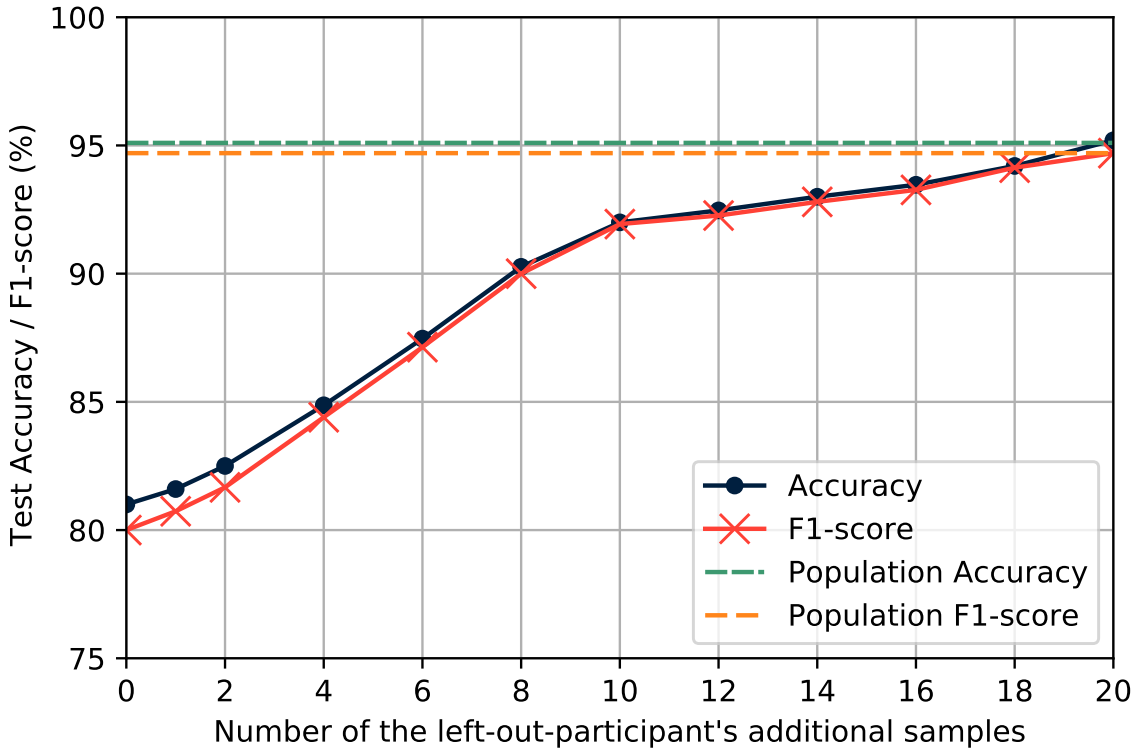


Figure 4.10: Offline analysis: results with the leave-one-participant-out (LOPO) data plus the ignored participant's additional samples range from 1 to 20 while performing offline analysis

4.2.4 Real-Time Simulation

I performed a real-time simulation to demonstrate the practicality of UnifiedSense by streaming the testing data. All trained models constructed in the offline analysis are stored and used for the real-time simulation.

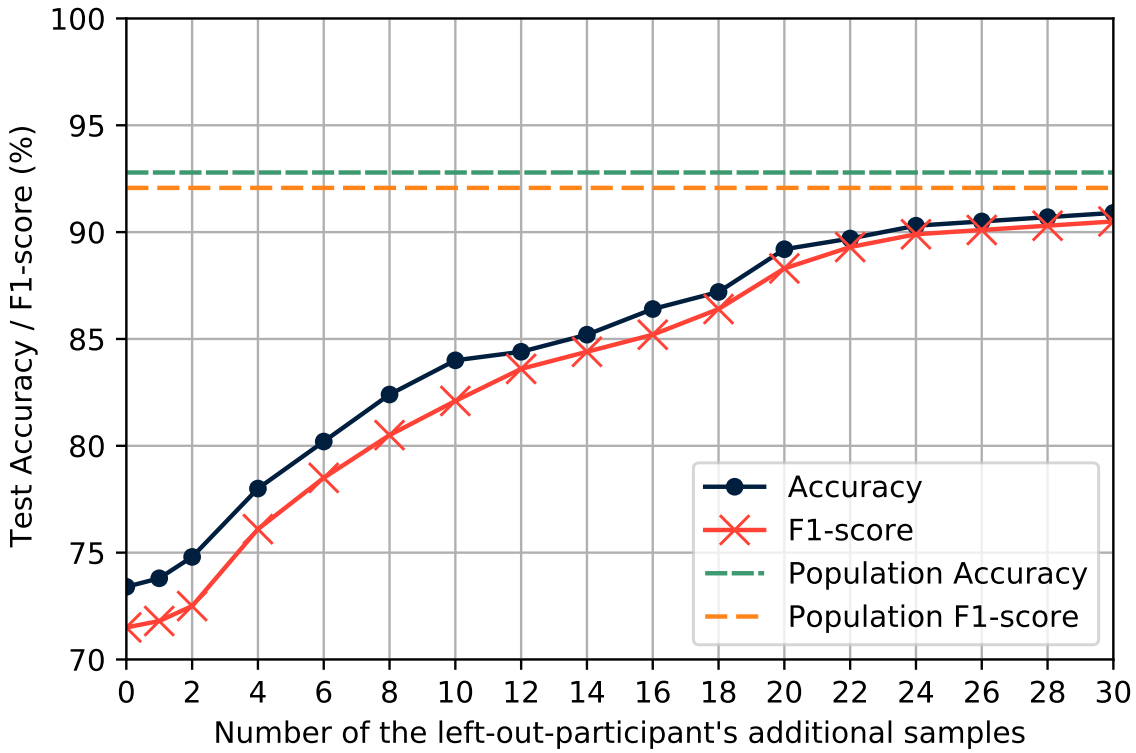


Figure 4.11: Real-time simulation: results with the leave-one-participant-out (LOPO) data plus the ignored participant's additional samples range from 1 to 30 while performing the real-time simulation.

A sliding window of 1600 ms was used for data streaming with a hop size of 400 ms. The data were streamed with a sampling rate of 100 Hz.

Population (within-subjects) Results

For each of the personalized models I constructed for the offline analysis, I loaded them into our real-time settings. Next, I streamed 50% testing data for each participant that I separated in offline analysis. These stream data were then fed to a real-time gesture detection and recognition setup. Note that these 50% data are continuous data with negative samples. UnifiedSense achieved 89.2% ($SD = 3.2\%$) of accuracy and 88.3% ($SD = 3.87\%$) of F1-score, on average.

Leave-One-Participant-Out (between-subjects) Results

I performed a real-time simulation similar to the within-subject analysis by streaming the LOP's data and feeding them into the real-time gesture detection and recognition settings. While performing the simulation for each participant, the corresponding stored model was loaded and used during real-time simulation. I stored all the trained models for each participant during an offline analysis in between-subjects conditions.

Finally, this gave us an overall accuracy of 73.4% ($SD = 1.5\%$) (F1-score: 71.5% ($SD = 1.89\%$)), a 15.8% drop in accuracy from the best-trained model within-subjects.

Further, I simulated the incorporation of additional samples for the ignored participant that I reported in Figure 4.10 in the case of offline between-subjects analysis. In the real-time simulation, I loaded the corresponding model that I saved earlier in offline between-subjects analysis and then streamed the LOP's data and fed them into the real-time settings. Figure 4.11 illustrates how the inclusion of a small amount of data from the user per gesture can improve the model performance in real-time simulation. The improved average accuracy and F1-score are reported in the same figure for LOPO using additional samples ranging from 1 to 30. It is evident from the figure that, with just ten gestures, the performance accuracy and F1-score improved from 73.4% to 84.01% and 71.5% to 82.1%, respectively. The performance approached near the population test accuracy beyond the inclusion of 22 additional samples, reaching 90.9% ($SD = 1.9\%$) accuracy and 90.5% ($SD = 1.6\%$) F1-score, which is also promising in real-time simulation.

4.2.5 Handling False Positives

In this section, I report the performance of the binary classification model in the UnifiedSense pipeline in rejecting false positives. The non-gesture examples were generated from the collected negative data by setting a 1600-ms sliding window with a step size of 50 ms. Then, I mixed all participants' gesture and non-gesture samples. With a 10-fold cross-validation using 70-30 train-test split, our binary classification model achieved an average accuracy of 99.8% ($SD = 0.53\%$) and an average F1-score of 99.9% ($SD = 0.42\%$). Moreover, the false-positive rate is 1.4 times per hour. Figure 4.12 reports the confusion matrix for the binary classification on the best model. It is evident from the figure that all the gesture examples are correctly classified as gestures. On the contrary, a small amount of non-gesture examples are being classified as gestures ($<1\%$), thus occurring false activations.

4.2.6 Gesture Performance Without Primary Device

The use-case of UnifiedSense hinges on a user being able to use a smart device even when that device is not physically present. Our offline and real-time evaluations have demonstrated promising results in recognizing gestures without using the data from the primary device. However, since the model is trained using the data collected while users are wearing the primary device, the model's performance may be affected if users wear a different wearable device or do not wear any device instead of the primary device. For instance, the way a user taps and the measurements captured by secondary devices may vary when tapping on a different watch or directly on their skin, compared to tapping on a smartwatch (primary device).

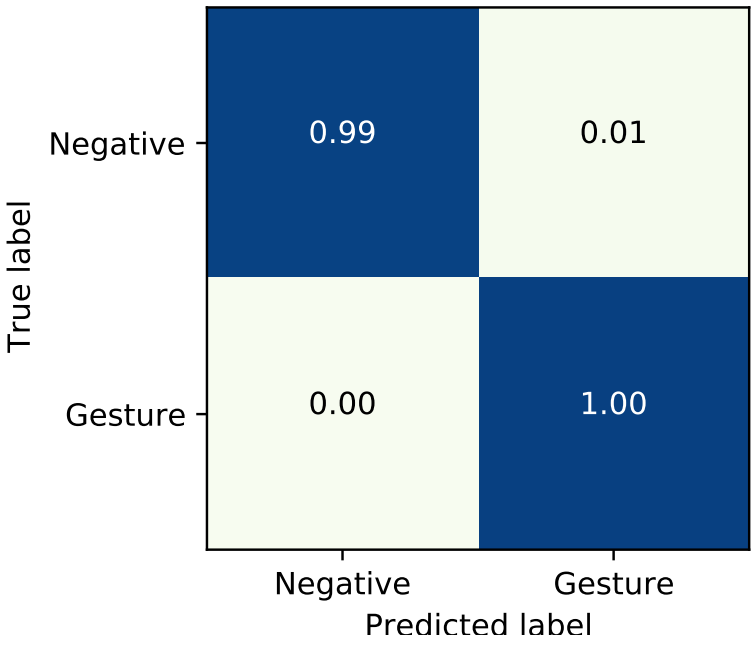


Figure 4.12: Confusion matrix of binary classification between gestures and negative data.

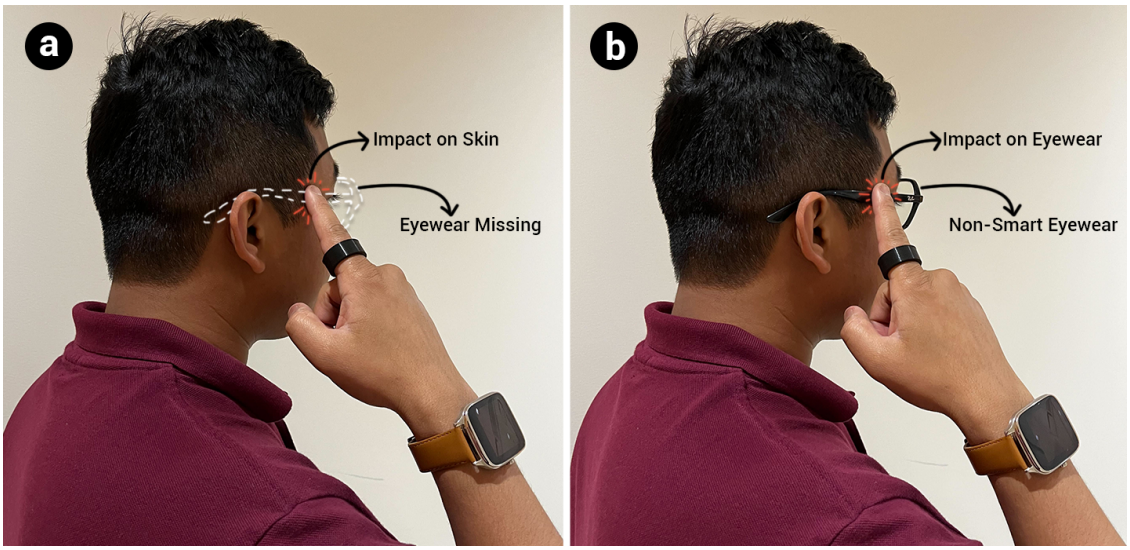


Figure 4.13: (a) Soft-hit profile and (b) Hard-hit profile.

I conducted a study to understand how the performance changes when the user wears a different device, which I call a hard-hit profile, where the user's finger collides with the physical object (e.g., Figure 4.13b), and when the user does not wear any devices, which I call soft-hit profile, where user's finger would collide with the user's skin (e.g., Figure 4.13a).

Participants

I recruited 6 participants ($Male = 5$, $Female = 1$) from a local university, who also participated in the first user study. I intentionally recruited participants who participated in the first user study to have a better comparison. All participants were right-handed, and their age range was between 23 to 35 ($Mean = 29.2, SD = 4.02$).

Procedure

Akin to the first user study, the researcher helped participants to wear all the smart devices except the primary device for both C1 and C3. The participants were instructed to perform touch gestures on "the right side of the forehead" for C1 and on "the wrist flesh" for C3, assuming the smart glasses and the smartwatches were present in the corresponding locations, respectively. The participants were asked to perform each gesture for 20 times consecutively in each configuration. Like the first user study, the participants were asked to reset their pose before performing each gesture. In this way, I collected the soft-hit profile data for both configurations. In addition, I also collected hard-hit profile data for evaluation by asking participants to wear non-smart devices in place of smart devices. For example, the participants wore regular sunglasses for C1 and wore an analog wristwatch for C3 to substitute smart glasses and smartwatches, respectively. This hard-hit profile data on these devices were collected to investigate the efficacy of our model even if users forget to wear smart devices and instead wear non-smart devices. Similar to the soft-hit profile, the participants were asked to perform each gesture 20 times in each configuration during hard-hit profile data collection. In total, I collected $2,400$ gesture samples in both configurations and both profiles (2 configurations $\times$ 6 participants $\times$ 5 touch gestures $\times$ 20 samples $\times$ 2 profiles). It took 25 minutes on average to complete the study. The participants were compensated with $10 each for their time.

Results

Using the best-personalized trained model, I tested the performance of the collected gestures from the third study for both soft-hit and hard-hit profiles. I saw that the model achieved an average accuracy of 92.8% ($SD = 3.7\%$) and an average F1-score of 91.95% ($SD = 5.3\%$) for the C1 configuration in the case of a hard-hit profile. On the contrary, the model achieved an average accuracy of 91.2% ($SD = 2.3\%$) and an average F1-score of 90.5% ($SD = 3.3\%$) for the same configuration in the case of a soft-hit profile. Similarly, for C3, the model achieved an average accuracy of 91.5% ($SD = 1.4\%$) and an average F1-score of 90.4% ($SD = 3.6\%$), and an average accuracy of 89.8% ($SD = 3.1\%$) and an average F1-score of 89.6% ($SD = 2.7\%$) for the hard-hit profile and the soft-hit profile, respectively. Figure 4.14 - Figure 4.17 report the confusion

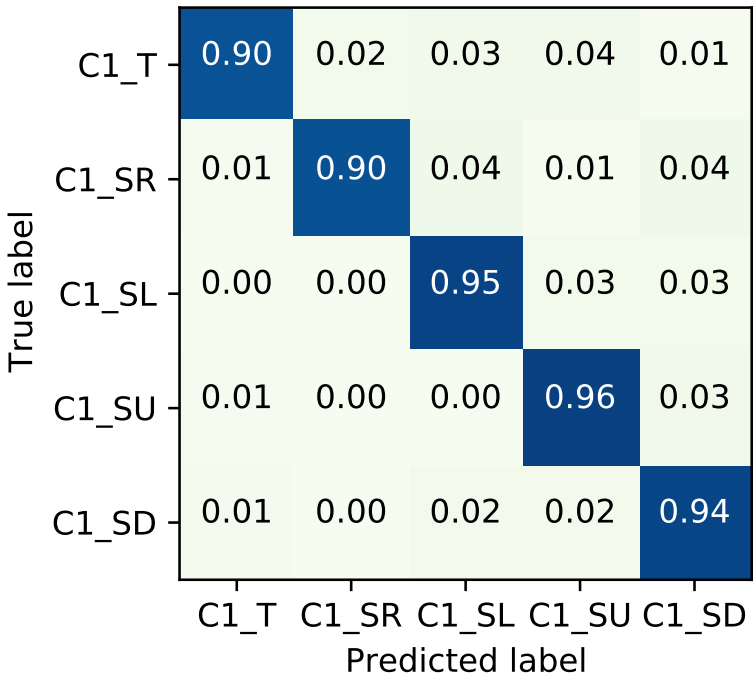


Figure 4.14: CM for C1 (hard-hit).

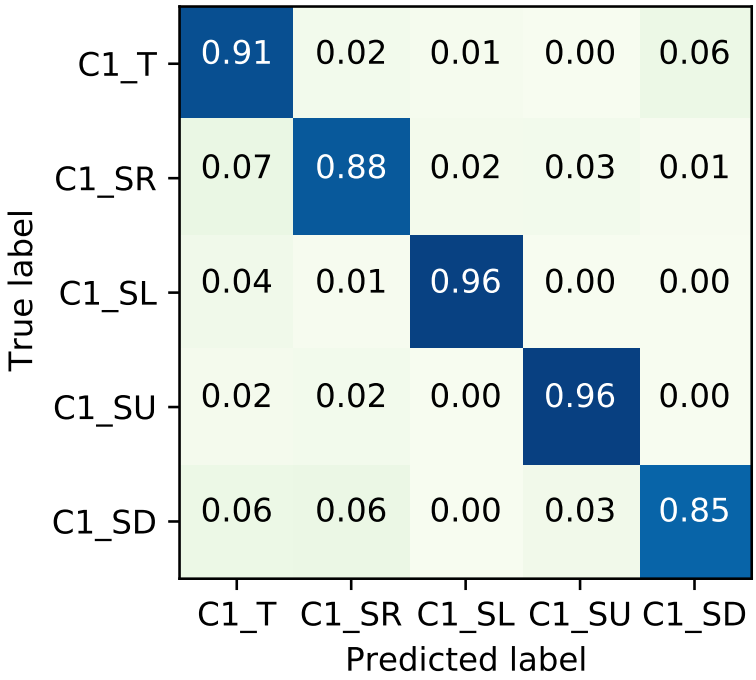


Figure 4.15: CM for C1 (soft-hit).

matrix for the configurations C1 (hard-hit profile), C1 (soft-hit profile), C3 (hard-hit profile), and C3 (soft-hit profile); respectively.

It is evident from the results that the hard-hit profile is slightly better than the soft-hit profile in terms of performance (accuracy and F1-score). The hard-hit profile's accuracy and F1-score are particularly higher

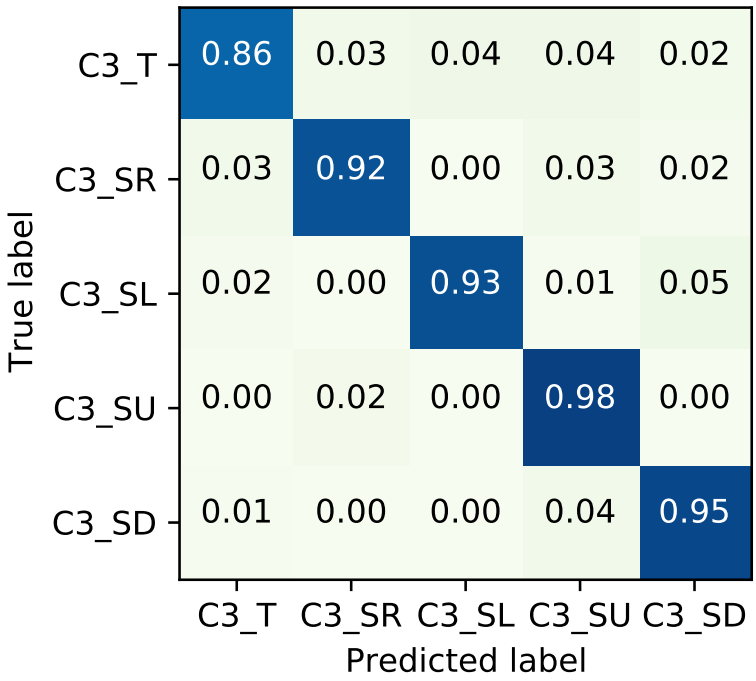


Figure 4.16: CM for C3 (hard-hit).

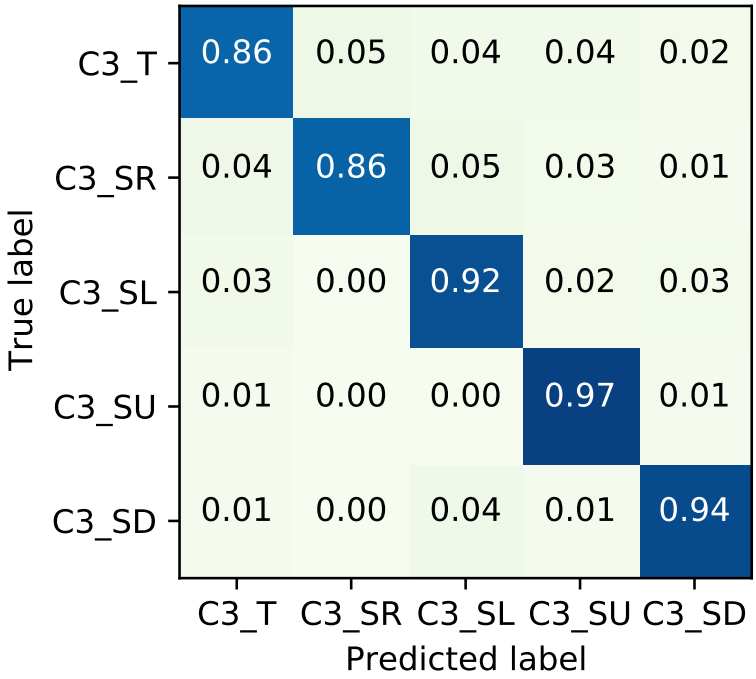


Figure 4.17: CM for C3 (soft-hit).

than the soft-hit profile by roughly 1%. It is obvious because the model trained for the UnifiedSense uses hard-hit profile data collected using the actual smart devices. In contrast, the testing data I used for evaluation were collected using hard-hit profiles (e.g., simulating a smart device with a non-smart one) and soft-hit profiles (e.g., performing gestures on skin surfaces such as wrist flesh). However, the performance of the

gestures in both of the profiles is comparable. UnifiedSense could be usable in practical situations, which can be interpreted from the confusion matrix (Figure 4.14 - Figure 4.17) with careful observations.

4.3 USense: UnifiedSense Dashboard

UnifiedSense utilizes a personalized gesture recognition model trained on gestures performed with the primary device. Our analysis indicates that the model can achieve a gesture recognition accuracy of over 90% after a few days of use. However, if the user changes the device configuration and the system fails to accurately recognize their input, it can lead to user frustration. To address this, I developed USense, an Android dashboard application that communicates the status of device connections and the gesture model to users.

Using USense, the user can access the list of available smart devices connected to their mobile device and their connectivity status (Figure 4.18a). USense also displays the supported gestures for each device (Figure 4.18b), with visualizations of the likelihood (as a percentage) of successfully detecting a gesture performed with and without the primary device. Additionally, users can test the gestures on USense and understand how the gesture was detected (Figure 4.18c).

While UnifiedSense can still function without USense, it provides a concise and intuitive interface for users to understand the connectivity, supported gestures, and performance of the UnifiedSense system, enabling effective cross-device interactions.

4.4 Discussion

One of the most distinctive characteristics of UnifiedSense is that the system gradually and automatically trains the model as the user continues using gestures over time, which is possible as the training data could be reliably labeled using the primary device input. As more wearable devices are being deployed and used by people, and more device-specific custom gestures are being used [202] to provide better functionalities, I anticipate that users will face the challenge of learning and remembering a large number of gestures. Also, as these devices offer efficient controls, their frustration can grow when they do not have the device. In such scenarios, I believe that UnifiedSense will allow users to continue using gestures that are familiar to them. Here, I discuss the feasibility and potential of UnifiedSense, the limitations of this work, and future work.

4.4.1 Feasibility and Potential of UnifiedSense

As UnifiedSense trains the gesture model as the user uses the gestures, the recognition performance will improve over time. However, if the configuration of wearable devices of a user is similar to that of other users,

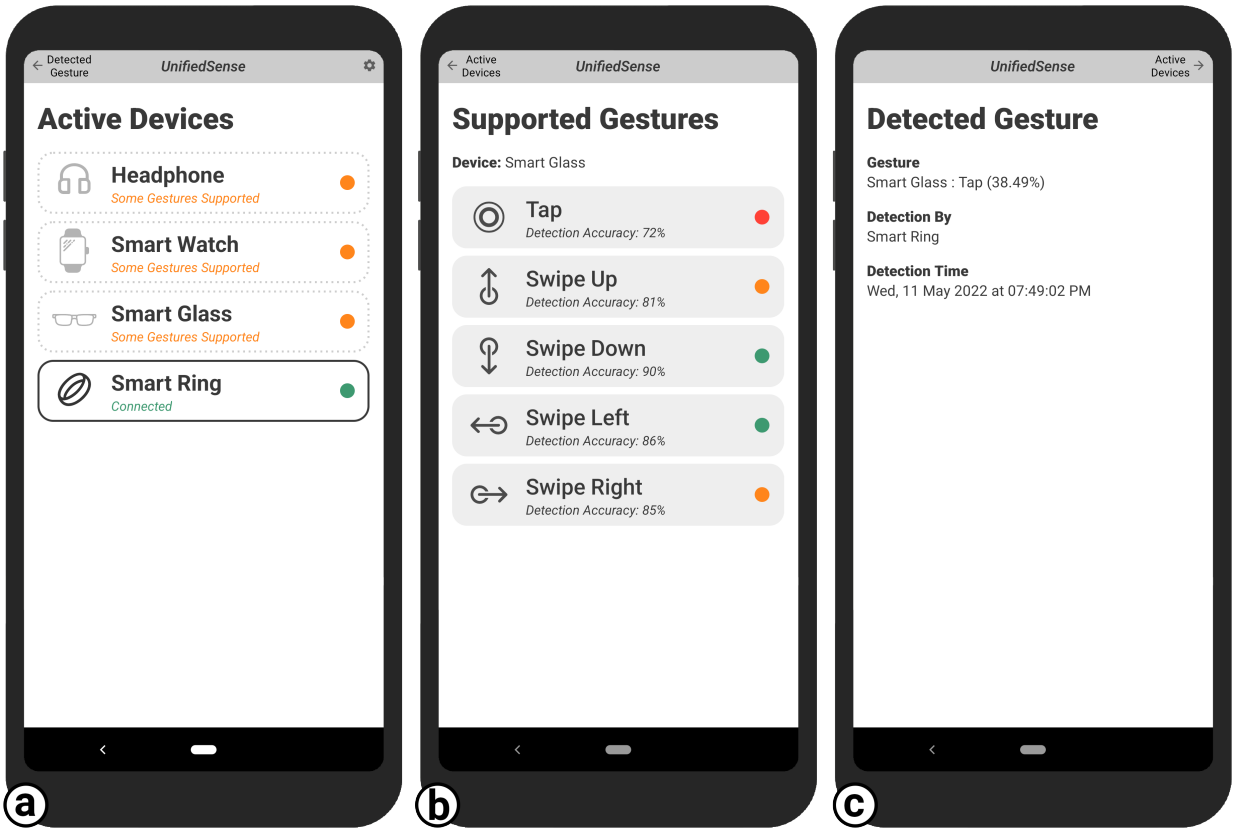


Figure 4.18: *USense* user interface: a) shows the connection status of the smart devices, b) shows the likelihood of correctly detecting gestures for each device, and c) shows the recently recognized gesture.

a model trained with others' data can help accelerate the training process. In the evaluation, I developed and evaluated our model using a LOPO method, simulating this scenario of using a model without any of the actual user's data. The results showed that UnifiedSense achieved 81.01% and 73.4% accuracy in offline analysis and real-time simulation, indicating that a generalized model could offer some level of recognition performance.

People have different ways of performing gestures, which could be the reason for the low performance of the generalized model. Our evaluation with a different number of user-specific training samples showed that just an additional eight training samples for each gesture could improve the gesture recognition accuracy to 84.01% for real-time simulation, and an additional 22 samples could achieve 90.9% real-time gesture recognition accuracy. Based on a study by Min et al., which reported that smartwatch users perform 95.6 interactions per day on average [292], I might expect that the model would be trained within a few days to be ready. This duration for additional training is longer than other methods, such as EarBuddy [37], which only requires 5 additional training samples to achieve 90% accuracy. However, different from methods that need the model to be completely trained for users to start using the gestures (e.g., [37]), in UnifiedSense, users can

immediately use the gestures on their primary devices. While the user is using the gestures, UnifiedSense automatically trains the model in the background. Therefore, the training process will not affect the primary device gesture recognition performance or user experience.

I believe UnifiedSense would enhance operating smart devices' usability with a unified gestural interaction technique because unified touch gesture recognition across the different configurations can improve gesture reusability [24] and interaction efficiency [293]. Moreover, UnifiedSense has the potential to improve user satisfaction in case a user forgets to wear a specific device, whereas our system facilitates the interaction technique for that device. To be precise, UnifiedSense does not substitute state-of-the-art interactions (i.e., touch gestures) on the primary device; rather, it creates the scope of alternative ways of interaction with the primary device, even if that device is absent.

UnifiedSense builds upon the assumptions that people will wear more wearables in the near future, and the problem of frustration will arise when they forget to wear smart devices or wear non-smart devices. It is innately difficult to design for a context that does not yet exist [294], in this case of designing for a potential future where usage of body-worn wearables is more far-reaching. A critical reflection on external validity for this work means looking forward to the projective validity [294] – do our results hold valid as more wearables become widely adopted and worn?

I anticipate that UnifiedSense will open the door for new research as more smart devices become available and people are wearing them simultaneously. A centralized system is required to monitor all the devices and check their statuses. Our UnifiedSense system opens the door for this purpose and future research. I admit that there will be potential engineering and research challenges regarding scalability and generalizability when more smart devices will be available at once. However, keeping those challenges aside, I argue that the UnifiedSense system has the potential to be implemented beyond context, which requires further investigations.

UnifiedSense hints towards *"enabling proactive cross-device interactions"* [146]. For instance, it is *users* who are "doing the ubicomp" [295], and Dourish argues that *users*, not designers, appropriate technology and thus create meaning for their interactions [296]. Therefore, instead of blending devices and hiding the boundaries between them, designers should embrace and leverage the heterogeneity and flexibility of devices and their "seams" [297, 295] – ultimately creating an ecology of devices that builds the conceptual foundation of cross-device computing [146]. The current design of UnifiedSense creates an ecology of connected devices and thus hints toward a proactive cross-device interaction.

4.4.2 Promoting Uniform Gestures Across Applications

While our focus in UnifiedSense lies in enabling device-independent gesture recognition, I acknowledge that the problem of inconsistent gestures can extend beyond device boundaries to different applications running on various platforms.

By incorporating a provision to integrate other recognizers like UnifiedSense into application design, developers can establish a standardized approach to gesture recognition. If all applications support the incorporation of uniform gesture recognition systems, it becomes possible to create a cohesive ecosystem where users can rely on consistent gestures across different applications.

It is important to note that UnifiedSense provides a framework for device-independent gesture recognition, and its impact on applications relies on the willingness of app designers to adopt such a system. If an app designer chooses to use their own recognizer, our methods have no direct influence. However, by encouraging the adoption of recognizers like UnifiedSense across the application landscape, I can promote a harmonized user experience with uniform gestures, enhancing usability and reducing the cognitive load associated with learning and remembering different gesture sets for various applications.

4.4.3 Low Accuracy Analysis

Achieving high accuracy in gesture recognition is crucial for the utility and usability of UnifiedSense. While our evaluation results demonstrated promising performance, there were instances where the recognition accuracy may be lower than desired. Several factors can contribute to the reduced accuracy, and understanding these limitations is essential for further improvements in the system.

As reported earlier in the chapter, the base model achieves a low accuracy of approximately 80% when used by a new user. This result could be attributed to the varied nature in which people perform gestures, which makes it challenging to develop a generalized gesture recognition model. This challenge can be more significant when trying to indirectly measure the gesture as there can be more factors affecting the measurements. Increasing the amount of user-specific training data and personalizing the model can improve the recognition accuracy. The main goal of UnifiedSense is to enable such training without explicit data collection sessions by making the system automatically collect the training samples as the user continues to use gestures.

4.4.4 Comparison to Other Recognizers

UnifiedSense utilized the TST model for gesture recognition. The reason for choosing the TST model was based on its reported state-of-the-art performance in various datasets, particularly in high-dimensional time-series data. While the gestures used in the study themselves were simple, the dimensionality of the

data is very high due to the use of multiple secondary devices. Given the high dimensionality of our dataset consisting of 24 tuples from 3 multiple channels, the TST model was deemed suitable for our purposes. Since the main focus of UnifiedSense is on the demonstration of the idea of automatically training the gesture recognizer while the user uses the gestures on a primary device and recognizing the gestures without primary devices, achieving the optimal performance was out of our scope. Future research could involve comparing the performance of different recognizers, including simpler approaches, in the context of UnifiedSense to gain further insights and optimize the gesture recognition system.

4.4.5 Limitations and Future Work

- In building UnifiedSense, I only considered IMUs, while wearable devices are equipped with an increasing number of sensors. I believe that utilizing multiple types of sensors would make the unified gesture recognizer support more diverse types of gestures at a better recognition performance. However, given the differences in the measurements, this will need further investigation.

- I only tested four smart wearable devices, a smart ring, a smartwatch, smart glasses, and smart headphones, while there are more diverse types of wearable devices available, such as earbuds, smart pendants, and smart footwear. The different placements of such devices would have different impacts on the recognition, which would require further study.

- The recognition of composite gestures (e.g., double-tap), which involve combining multiple individual gestures, may pose additional challenges. The majority voting approach I used in the real-time implementation pipeline may limit the construction and accurate recognition of composite gestures. Investigating and implementing more sophisticated techniques, such as sequence modeling or hierarchical recognition, could address this limitation and improve the system's ability to handle composite gestures effectively.

- While UnifiedSense achieved comparable gesture recognition performances with other wearable-based gesture recognition systems (e.g., [202, 169]), I think that the performance could be further improved by using a more careful tuning of the deep learning model and by introducing few-shot learning [202].

- I used *USense* only to inform the user of the system's current status. Future work would expand its use to allow users to reconfigure gestures across devices and design custom gestures to increase the input vocabulary.

Chapter 5

Investigation of Hand-Movements Manipulation

Over the past few years, Virtual Reality (VR) has gained popularity. With the high-fidelity 3D visual display and 6-degree-of-freedom (DOF) tracking, VR systems are now used for various applications ranging from entertainment and education applications to productivity and work-related tools. Since VR is a unique computing platform that offers immersive experiences by separating the virtual and physical environment, it has also opened up many possibilities for enhancing user experiences by manipulating the users' perceptions, utilizing the dominance of the visual sense over other senses [298]. For example, slightly diverging the trajectory of the visual hand movement from the actual hand movement could redirect the user's hand to a limited number of haptic props and surfaces [299, 217, 218]. Manipulating the amount of visual hand movement for the same physical hand movement could create the illusion of weight change [220] and the illusion of object size change [300]. A study also investigated the use of dynamic movement gain to make VR interactions more ergonomic [301].

With the great potential to improve the immersion and usability of VR, these perception manipulation techniques have been intensively studied. Researchers have investigated the thresholds for the level of perception manipulation before participants notice the discrepancy between the real and virtual hands and break the body ownership [302, 303, 304, 305]. While promising, dynamic perception manipulation may also impose a potential side effect of affecting the skilled motor movement, which humans acquire by repeating the same movement for an extended period [306, 307, 308]. Humans rely on both visual and kinesthetic senses, and they learn to use the optimal hand movement based on the mismatch between the intended and the actual movement [227, 228, 229, 230]. The learning of optimal motor movement requires repeated practice to

build muscle memory and makes users complete tasks involving motor movements faster, more consistently, and more stably with reduced attention demand [231]. With dynamic perception manipulation that changes the virtual-real hand mappings over time, it can be difficult for a user to build up muscle memory to reach the level of automaticity. However, there is little study about the potential effect of perception manipulation on motor skill learning.

In this chapter, I investigate the effect of two commonly used perception manipulation techniques for hand interaction – orientation and magnitude manipulation techniques, on hand movement and task performance during reaching tasks. My investigation is situated within the theoretical framework of Woodworth's two-component model, a construct that has been foundational since 1899 [309]. Grounded in Woodworth's two-component model, which delineates goal-directed movements as comprising an initial ballistic phase and a subsequent corrective phase for trajectory adjustments using visual cues [309]. Meyer et al. expanded on this by highlighting how the corrective phase merges visual and proprioceptive feedback, ensuring movements are both precise and coordinated [230]. This phase is crucial for proprioceptive adjustments, aiding in limb positioning and real-time corrections [230]. Research has shown a preference for proprioceptive feedback in motor task refinement and indicated that visual feedback alterations necessitate proprioceptive recalibration [310].

Goal-directed reaches are a foundational building block of everyday human motor behavior, and they have therefore been studied extensively in the motor control literature. Given that virtual hand reaches are effectively goal-directed reaching movements performed in VR, this body of work and its associated tools may be useful for understanding users' behaviors during virtual hand reaching. Moreover, kinematic analysis (KA) is a vital tool for understanding goal-directed reaches, and it has been extensively studied in motor control literature. With virtual hand reaches in VR resembling these movements, KA techniques, employed for over a century, offer insights into various aspects of such reaches (e.g., [311, 312, 313, 314]). These techniques involve analyzing motion-tracking data to quantify properties like speed, efficiency, and smoothness of the reaches. As modern VR systems easily capture this data, KA metrics can be applied to assess virtual hand reaches, proving particularly beneficial in applications such as monitoring arm function recovery in stroke rehabilitation or tracking learners' progress in motor skills training within VR environments (e.g., [315, 316, 317]). Understanding how perception manipulations might influence the kinematic properties of virtual hand reaches would bolster numerous research and design efforts at the intersection of human movement science and virtual reality. Considering this, I focused on analyzing the users' hand movement with kinematic metrics in this study.

By conducting two user studies focusing on orientation and magnitude manipulations in VR, I found that movement behaviors significantly altered while task completion times remained unchanged with manipulations, especially at higher manipulation levels. Notably, the proprioceptive quality, or ballistic movement, was

markedly affected, requiring participants to perform additional corrective actions to accurately reach targets. This necessitated quicker corrective movements to compensate for initial inaccuracies, maintaining overall task completion times but potentially increasing user effort and coordination [230]. Such adjustments could elevate cognitive load, impacting motor control strategies and possibly detracting from motor performance efficiency [230].

5.1 Method

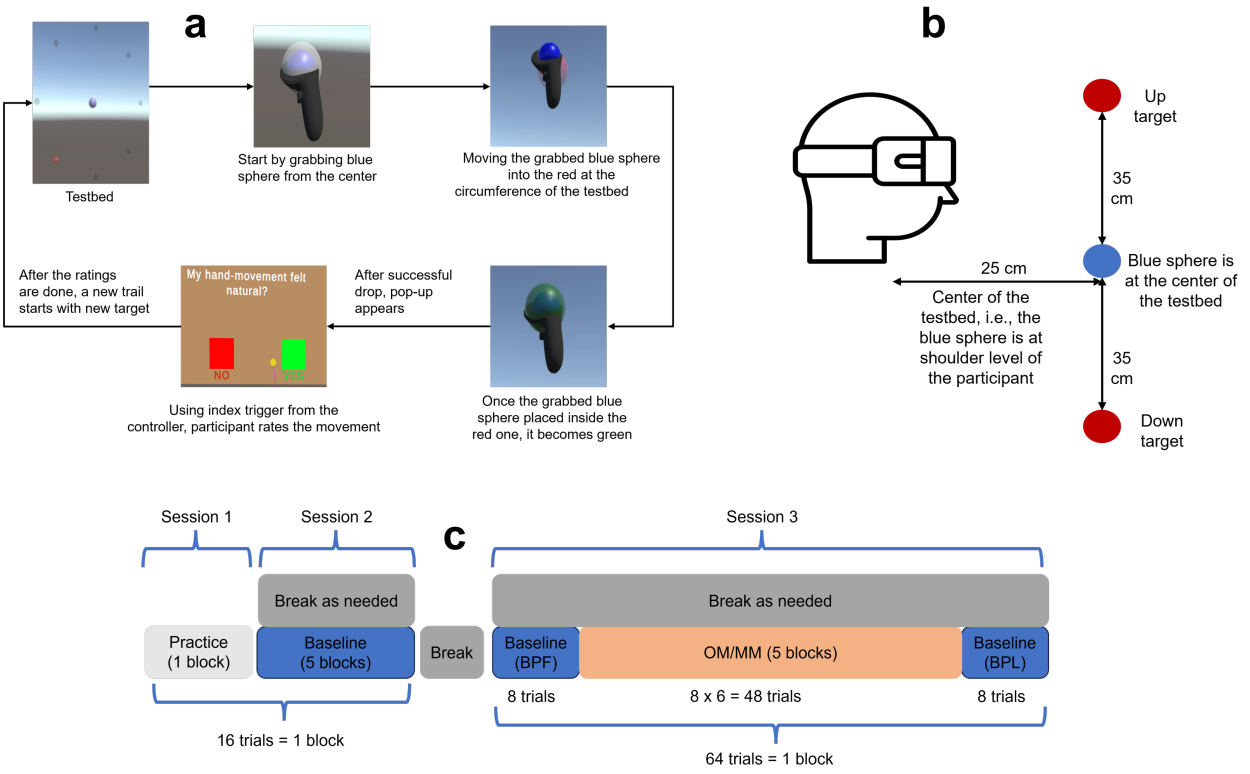


Figure 5.1: (a) The sequence of events for the performed task used in the study. (b) The distances between the VR user, the start position (blue sphere), and the target positions (red spheres) are shown. (c) The study sessions are shown as a sequence of blocks in a diagram. Here, BPF and BPL refer to "Baseline Padded First" and "Baseline Padded Last," respectively.

Two user studies were conducted to investigate the effects of perception manipulation on hand movements during object manipulation tasks in VR. The designed task required participants to grab an object, move it, and place it into a designated target position. This task essentially amalgamates elements of both target selection and point-to-point reaching tasks. The two studies conducted were (a) *angle redirection manipulation* (also referred to as Orientation Manipulation (OM)), and (b) *gain redirection manipulation* (also referred to as Magnitude Manipulation (MM)). Both studies maintained an identical experimental setup and design,

with variations solely in the applied manipulations. No same participant was enrolled in either study. The study protocol was approved by the Institutional Review Board (IRB).

5.1.1 Participants

OM Group: A total of 17 participants were recruited for the OM condition. All participants were right-handed (Male = 10, Female = 7). The group's average age was 25.7 years (SD = 3.8). Participants were asked to provide their experiences in using VR through a short questionnaire before the study. Within this group, seven participants reported prior VR experience for gaming, four for research studies, two had never used VR, and the remainder used VR occasionally. One participant could not complete the tasks within the allotted time due to impaired depth perception and was subsequently removed from the study.
MM Group: For the MM condition, 17 right-handed participants were recruited (Male = 7, Female = 10). The average age was 23 years (SD = 2.8). Nine participants had previously used VR for gaming, two for research studies, five had no prior experience, and the rest used VR occasionally.
Participants in both groups were recruited through mailing lists, Slack channels of student groups, and word-of-mouth referrals. Each participant received a $20 gift card as compensation for their time.

5.1.2 Study Design and Procedure

The study employed a within-subjects design and was conducted in a controlled lab environment. All participants were provided with an *Oculus Quest Pro* VR headset and *Controllers*. The participants sat in a comfortable stationary chair that was secured to the floor. The experimenter sat at a desk to the side of the participants and monitored their performance. They were briefed that the study focused on "usability", disguising the actual research purpose to mitigate expectation biases.

In each task, participants were asked to grab a blue sphere located at the center of their view by using the trigger button and place the blue sphere inside the red target sphere (Figure 5.1a) as quickly and accurately as possible. If the blue sphere was not correctly placed inside the red sphere, the trial was considered incomplete and the participant was asked to try it again. After the trial, the participants were asked to answer if their hand movement felt natural or not, using the buttons shown on the VR display. After answering the question, the participants started the next trial.[1] The participants were also instructed to rest as needed between trials to minimize any effects of fatigue.

[1]In the manipulated trial, the manipulation was initiated the moment participants grabbed the blue sphere and persisted until they released it inside the red sphere. Upon completion of this task, the controller was rendered invisible for a 0.5-second duration to facilitate the repositioning to its original, unmanipulated state. This temporary invisibility was strategically employed to ensure that the repositioning occurred without the participants' awareness. Concurrently, a pop-up window was displayed at the point of task completion.

The position and size of the objects were defined using Unity units (1 unit ≡ 1 meter). At the beginning of each session, the experimenter calibrated the VR headset to ensure that the location of targets relative to participants would remain consistent across all participants and sessions. The size of the red spheres was 3.5 cm in radius and the size of the blue sphere was 2.5 cm in radius. The testbed with these nine spheres was placed on a surface in front of the VR user, which is 25 cm away from the shoulder level of the VR user (Figure 5.1b). The distance between the blue and red spheres was 35 cm. The targets in the circumference were placed 45° apart from each other.

The whole study was divided into three sessions (Figure 5.1c). The first session served as a practice round and consisted of one block that included 16 trials across eight different target orientations. No manipulations were applied during this practice session. I moved to the actual study as soon as they completed the practice and felt comfortable with the settings. The data collected during the practice session were discarded from the analysis.

In the second session, baseline data were collected. This session comprised five blocks, each containing 16 trials to ensure consistency. Within each block, every target orientation was represented in exactly two trials, resulting in a total of 80 trials per participant for this baseline session. Breaks were provided between blocks as necessary to minimize fatigue, and participants were given a 2-3 minute break and asked to take off the VR headset at the end of this session. The target order was randomized in this session.

The third and final session introduced manipulation trials. Like the baseline session, these sessions also had five blocks each. However, each of these blocks contained 64 trials. The first and last eight trials within each block were not manipulated to serve as control data, whereas the remaining 48 trials were manipulated across six levels and eight target orientations. Non-manipulated trials were intentionally placed at the beginning and end of each block to mitigate any biases; I refer to these baselines as *Baseline Padded First* (BPF) and *Baseline Padded Last* (BPL), respectively. A cumulative total of 320 manipulated trials were collected from each participant during this session. Participants were given two-minute breaks between each block to rest. Given the higher number of trials in these blocks, participants were also asked to rest within each block as needed. Both the target and the level selection were randomized for the manipulated trials. The six different levels of manipulation for the OM group were ±18°, ±12°, and ±6°. The angle for the baseline level is 0°. Similarly, six scaling levels were employed in the MM group: 1.0±0.30, 1.0±0.20, and 1.0±0.10, with a baseline scaling factor of 1.0. Equation 5.1 and Equation 5.2 were used to calculate the manipulated hand positions during the trials, respectively, for the OM and MM. Here, θ represents the applied angles for OM, whereas $scaling\_factor$ represents the applied scales for MM. The manipulations were only applied on the XY plane, which can be observed from the equations. I did not apply manipulations on the Z-axis (i.e., the depth) because the participants are sensitive to depth sensing, which was found in the study by Benda et

al. [305].

In Equations 5.1 and 5.2, ΔX, ΔY, and ΔZ represent the change in hand position along the X, Y, and Z axes, respectively, calculated as the difference between the current and previous hand positions. The angles ΔX_Angle and ΔY_Angle are derived from the original changes in position (ΔX, ΔY) adjusted by the rotation angle θ, facilitating the manipulation of hand movement in the VR space. The updated current hand position is then recalculated by applying these angular adjustments to the original positions, effectively simulating the hand's movement within the virtual environment. This approach allows for a nuanced control and manipulation of hand movements in VR.

For the OM group, the total number of trials collected was: (a) baseline session: 16 *participants* × 5 *blocks* × 16 *trails* = 1280, and (b) manipulated session: 16 *participants* × 5 *blocks* × 64 *trails* = 5120. Similarly, for the MM group, the baseline session had 1360 trials, and the manipulated session had 5440 trials.

$$\left.\begin{aligned}
\Delta X &= current_hand_position_X - previous_hand_position_X \\
\Delta Y &= current_hand_position_Y - previous_hand_position_Y \\
\Delta Z &= current_hand_position_Z - previous_hand_position_Z \\
\Delta X_Angle &= \Delta X \times \cos\theta - \Delta Y \times \sin\theta \\
\Delta Y_Angle &= \Delta X \times \sin\theta + \Delta Y \times \cos\theta \\
current_hand_position &= Vector(current_hand_position_X + \Delta X_Angle, \\
&\qquad current_hand_position_Y + \Delta Y_Angle, \\
&\qquad current_hand_position_Z + \Delta Z)
\end{aligned}\right\} \tag{5.1}$$

$$\left.\begin{aligned}
\Delta X &= (current_hand_position_X - previous_hand_position_X) \times scaling_factor \\
\Delta Y &= (current_hand_position_Y - previous_hand_position_Y) \times scaling_factor \\
\Delta Z &= (current_hand_position_Z - previous_hand_position_Z) \times 1.0 \\
current_hand_position &= Vector(current_hand_position_X + \Delta X, \\
&\qquad current_hand_position_Y + \Delta Y, \\
&\qquad current_hand_position_Z + \Delta Z)
\end{aligned}\right\} \tag{5.2}$$

5.1.3 Deception and Debriefing

Participants were initially informed that the study aimed to examine the usability of VR technologies. At the end of the study, a debriefing session was conducted. Participants were provided a printed debriefing statement clarifying the study's true objectives. They were also presented with a post-debrief consent form, inquiring about their permission to use their data now that the study's actual intent had been revealed.

5.1.4 Data Analysis

The experimental software captured the virtual controller's x-, y-, and z-position with the timestamp at a sampling rate of 72 Hz since the VR's refresh rate was 72 Hz. The software collected the actual virtual

Table 5.1: Kinematic Metrics in the Study.

Metric	Notation	Description
Movement Time	MT	Time from trial start to end in seconds. Higher values signify less efficient movements.
Primary Submovement Time	T1	Duration of the initial submovement.
Secondary Submovement Time	T2	Duration of subsequent submovements. Related to Movement Time as $MT = T1 + T2$.
Peak Velocity	PV	Max speed of the controller in cm/s. Higher values denote faster movements and occur during the ballistic phase.
Primary Submovement Distance	PSD	Distance covered in T1. Greater values suggest more distance covered through ballistic movement.
Secondary Submovement Distance	SSD	Distance covered in T2. Greater values indicate a higher extent of distance covered through corrective movements.
Peak Velocity in Secondary Submovement	PV_{ss}	Max speed during T2 in cm/s. Higher values signify faster corrective movements.
Average Speed in Secondary Submovements	$AvgS_{ss}$	Average speed during T2 in cm/s. Higher values imply faster corrective movements.
Secondary Submovement Point-to-Point Distance	SSD_{p2p}	Euclidean distance from end of T1 to target. Greater values signify more distance covered via corrective movements and indicate a greater separation from the target after ballistic movement.

controller position data when the perception manipulations were applied. The experimental software also collected each trial's start and end timestamps. The start timestamp is when the participant grabbed the blue sphere; the end timestamp is when the participant dropped the blue sphere into the red sphere. The software did not log the unsuccessful trials, such as when the participants dropped the sphere incorrectly. Individual trial's positional data was segmented using these start and end timestamps. Missing position data were interpolated using spline interpolation, and the data were resampled to a constant 72-Hz sampling rate. Filtered position data were then used to calculate the cumulative Euclidean distance traveled by the controller for each trial and the resulting profiles were differentiated to derive the velocity and acceleration profiles used to calculate kinematic metrics reported in Table 5.1. The velocity profile was smoothed using a Gaussian kernel filter with a standard deviation parameter of 5 [318]. Similarly, the acceleration profile was smoothed using a Gaussian kernel filter with a standard deviation parameter of 4 [318]. I used the smoothed velocity and acceleration profiles to segment the primary and secondary submovements, a similar approach leveraged from [319].

The effects of movement time, peak velocity, speed, and distance traveled during different submovements were assessed using a one-way repeated-measures ANOVA. Significant differences involving more than two means were examined using Tukey's HSD post hoc comparisons ($p < 0.05$).[2]

[2]p-value annotation legend in the reported figures (Figure 5.3 - Figure 5.13): $*:\ 1.00e^{-02} < p <= 5.00e^{-02}$, $**:\ 1.00e^{-03} < p <= 1.00e^{-02}$, $***:\ 1.00e^{-04} < p <= 1.00e^{-03}$, $****:\ p <= 1.00e^{-04}$, and $ns: p >= 0.05$; ns are not marked in the figure.

5.2 Results

5.2.1 Detection of Perception Manipulation

Figure 5.2 displays participants' reports of hand movement unnaturalness across both studies, with baseline conditions (B, BPF, BPL) set at an angle of 0° and a scale factor of 1.0. Participants more frequently perceived movements as unnatural with increased manipulation intensity. In orientation manipulation, unnaturalness perceptions were similar for removedboth clockwise and anti-clockwise adjustments (Figure5.2a). For magnitude manipulation, the ratio of feeling the movement unnatural was higher when the manipulation reduced the hand movement than when the manipulation increased the movement by the same ratio (Figure 5.2b).

Applying a 50% threshold to detect noticeable unnaturalness, orientation manipulations were perceptible between ±10° to ±12°, slightly exceeding findings by Zenner and Kruger [303]. This discrepancy might be attributed to participants being unaware of the study's true aim, potentially reducing their sensitivity. For magnitude manipulations, sensitivity was greater for scaling down (threshold at 0.8 or higher) compared to scaling up (threshold above 1.2, nearing 1.3), consistent with observations by Benda et al. [305].

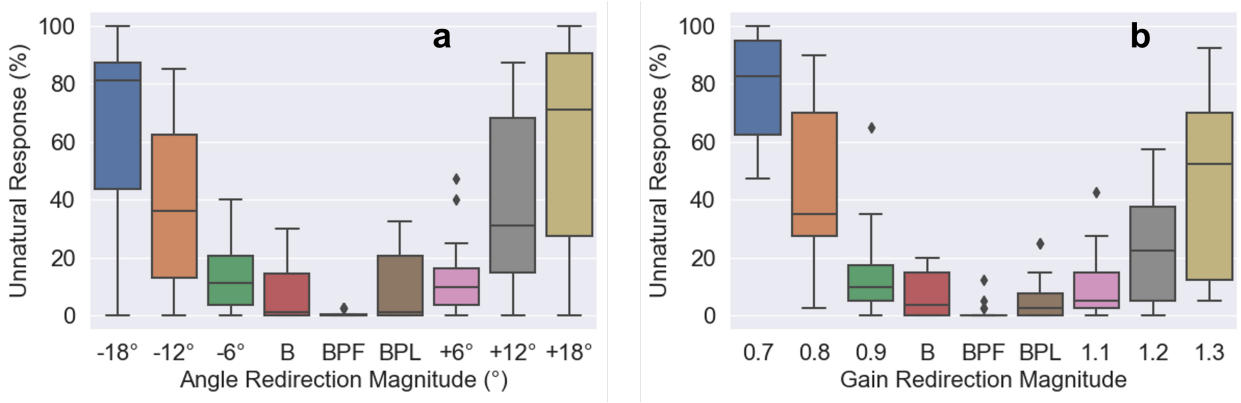


Figure 5.2: Percentage of "Unnatural" responses for (a) Orientation Manipulation, and (b) Magnitude Manipulation.

5.2.2 Level-Wise Analysis

Impacts in Kinematic Metrics for OM

In this investigation into the effects of orientation manipulation within VR, my findings illustrate the adaptability of participants for manipulations. Despite manipulations, task completion times remained consistent across different manipulation levels, underscoring the participants' capacity to swiftly adapt to VR manipulations (Figure 5.5 (a)).

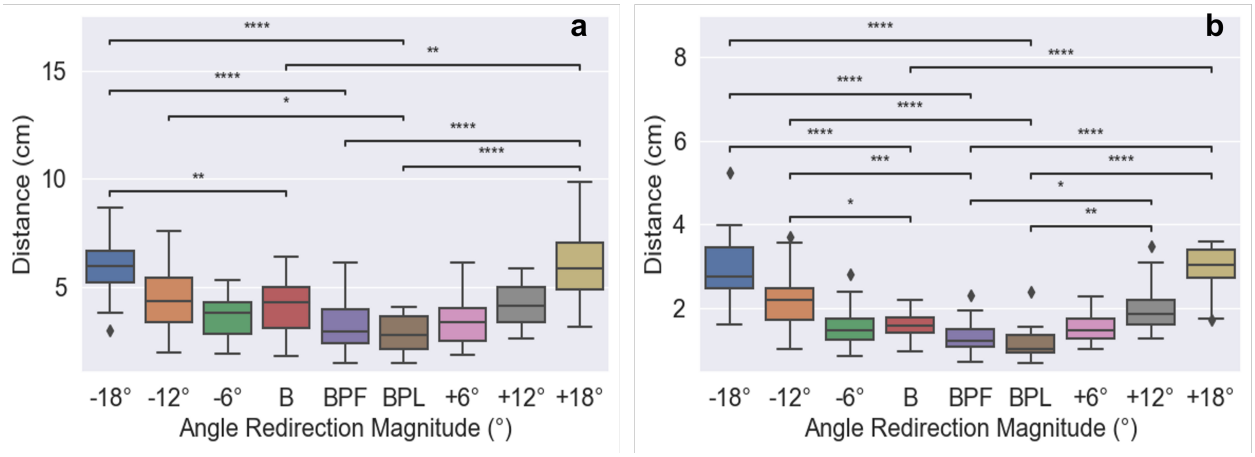


Figure 5.3: Level-wise – (a) SSD and (c) SSD_{p2p} for OM study.

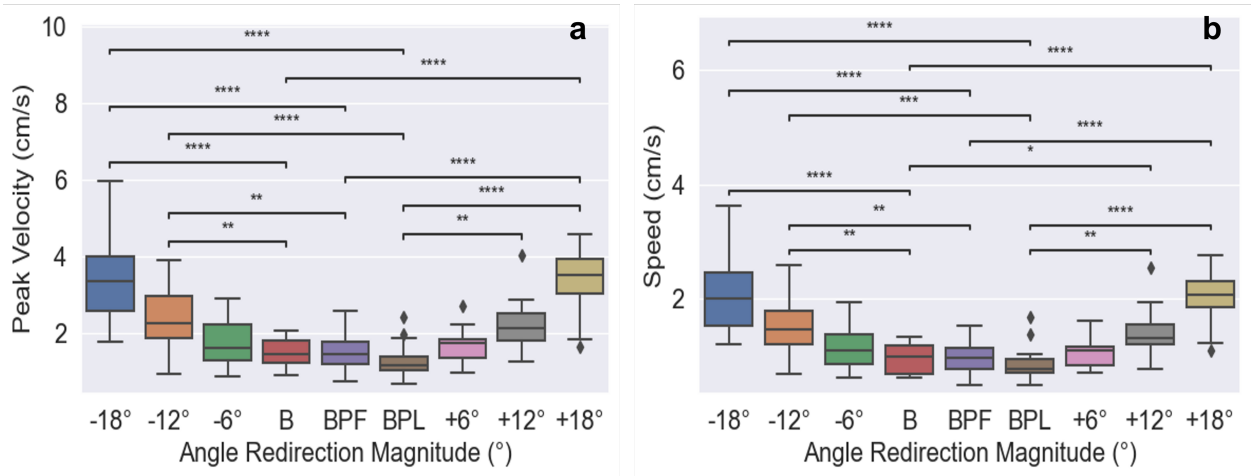


Figure 5.4: Level-wise – (a) PV_{ss} and (c) $AvgS_{ss}$ for OM study.

Analysis of the primary (T1) and secondary submovement times (T2) revealed no significant differences in response to the manipulations, indicating a stable approach toward the target throughout the experiment. This stability suggests robustness in participants' initial movement strategy, unaffected by the VR manipulations introduced.

Further examination into the distances covered during these movements unveiled more nuanced effects. While PSD showed little change, the SSD and point-to-point SSD (SSD_{p2p}) increased with greater levels of manipulation. This increase in SSD and SSD_{p2p} highlights how participants compensated for the VR manipulations, undertaking larger corrective movements as the discrepancy between expected and actual virtual environments grew (Fig 5.3).

On the other hand, PV metrics did not exhibit any significant differences across manipulation levels. However, in the case of corrective actions, both the peak velocity during secondary submovements (PV_{ss}) and

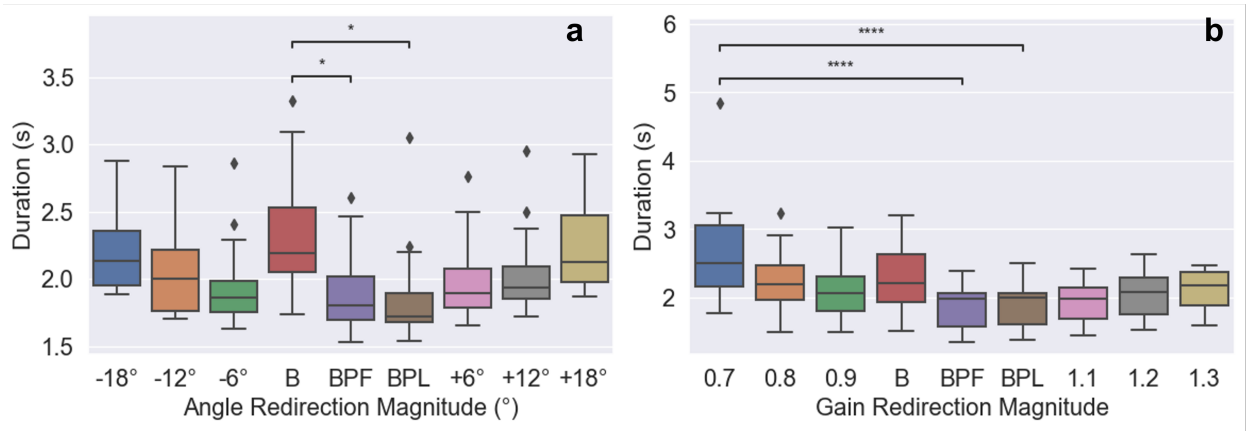


Figure 5.5: Level-wise – (a) MT for OM study and (c) MT for MM study.

the average speed during these movements ($AvgS_{ss}$) escalated with higher levels of manipulation (Fig 5.4).

Impacts in Kinematic Metrics for MM

In the case of MM, the participants' MT remained consistent compared to the baseline condition except for the scale factor of 0.7. This showed us that extreme manipulations make the participants move differently (Figure 5.5 (b)).

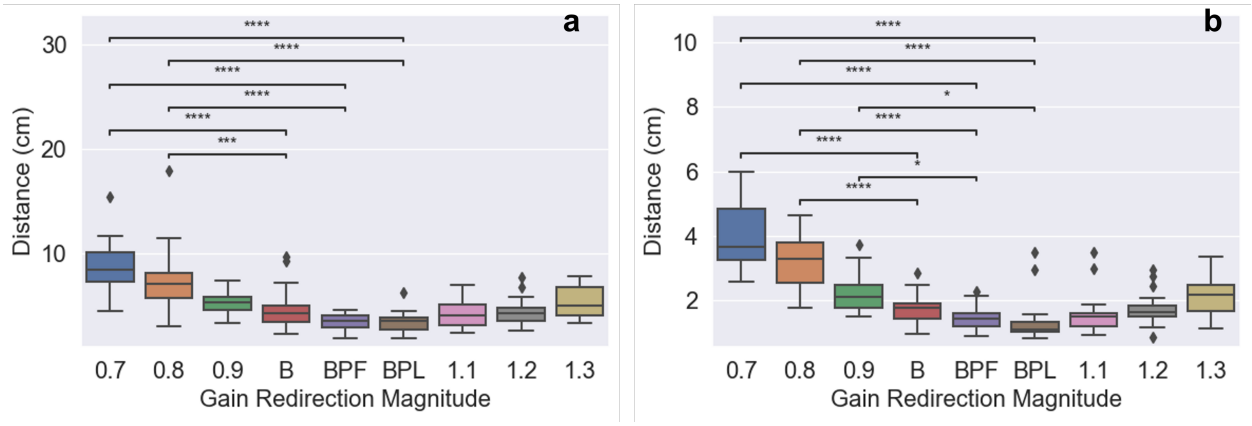


Figure 5.6: Level-wise – (a) SSD and (c) SSD_{p2p} for MM study.

Observing movement time during ballistic movement (T1) and subsequent corrections (T2), a similar pattern emerged. With the exception of the extreme condition (scale = 0.7), participants' initial movements and subsequent adjustments remained relatively consistent across MM. This indicates that magnitude manipulation beyond certain thresholds might degrade users' performance during corrections. Interestingly, while assessing the distance traveled by participants during these tasks (PSD), I observed minimal differences across most conditions, except for instances when objects appeared significantly closer than usual (scale down).

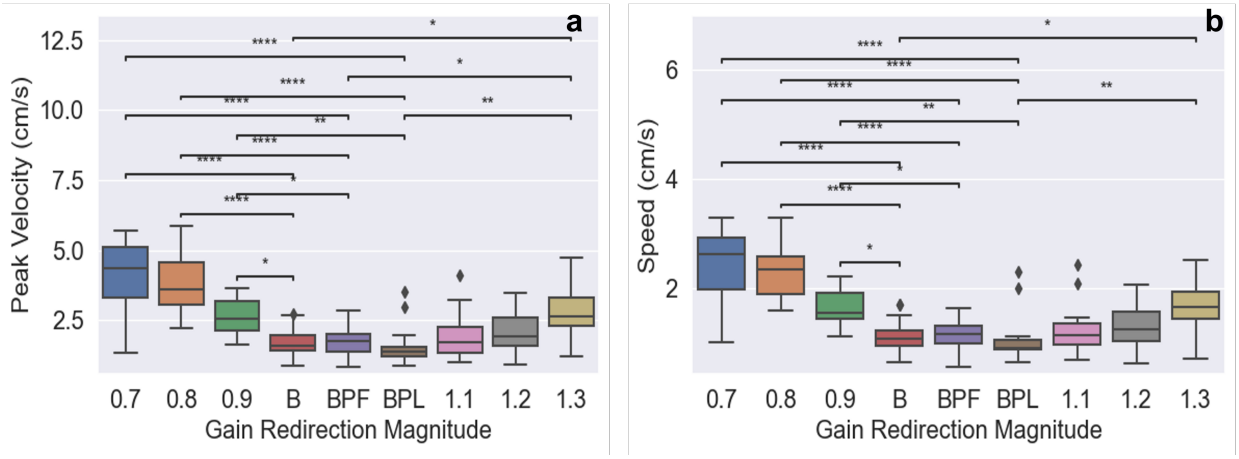


Figure 5.7: Level-wise – (a) PV_{ss} and (c) $AvgS_{ss}$ for MM study.

This consistency suggests that our ability to estimate distance in VR remains relatively steady unless the manipulation is quite extreme (Fig 5.6). Upon delving deeper into how participants adjusted their movements (SSD) and aimed for precision (SSD_{p2p}), I noted that creating the illusion of objects being closer (scale down) posed considerable challenges. Participants made more substantial corrections and exerted greater effort to achieve accuracy, especially when VR compressed distances (Fig 5.6). Lastly, when examining the speed of participants' corrective movements (PV_{ss}) and their average speed ($AvgS_{ss}$), I observed significant increases with higher levels of VR manipulations, particularly during scale-down manipulations. This heightened speed was most evident during scale-down manipulations since participants were required to move farther to reach the goal. (Figure 5.7).

5.2.3 Block-Wise Analysis

Delving further into our study's block-wise analysis unveils insights into participants' responses, particularly when exposed to orientation and magnitude manipulations over time. For this analysis, kinematic metrics, as detailed in Table 5.1, are plotted against block numbers. The graph depicts the first five blocks from the baseline (session 2), followed by the subsequent five blocks from the manipulation phase (session 3). Additionally, session 3 incorporates baseline measurements, both at the beginning and end.

Impacts in Kinematic Metrics for OM

Initially, in the baseline phase (first five blocks), participants were getting accustomed to the VR environment, with the first two blocks showing noticeable differences in movement time (MT) compared to later blocks (Figure 5.8a). This suggests an initial learning curve as participants adjusted to their hand movements.

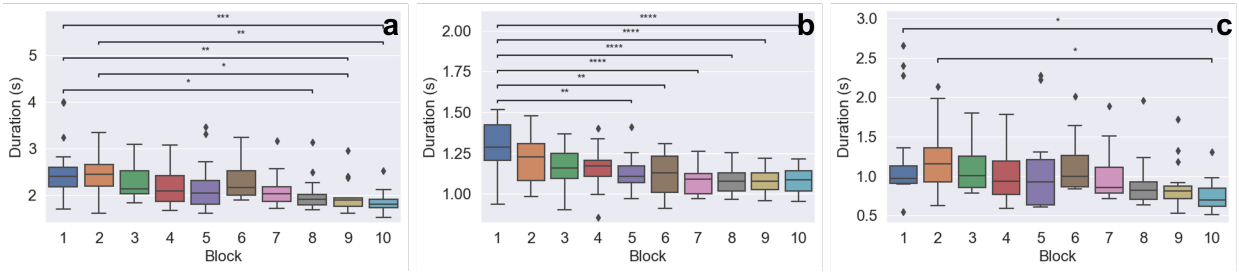


Figure 5.8: Block-wise – (a) MT, (b) T1, and (c) T2 for OM study.

As I transitioned into the manipulation phase (last five blocks), a significant adaptation process was observed. Despite the introduction of perception manipulations, participants demonstrated adaptability, as evidenced by the consistency in movement time (MT) across different blocks. Analyzing both primary (T1) (Figure 5.8a) and secondary (T2) (Figure 5.8b) submovement times revealed a learning effect. Initially, participants took longer for ballistic movements (T1), but as they progressed, particularly beyond the baseline session, their speed increased, indicating optimization of their movement strategy to counteract the VR manipulations.

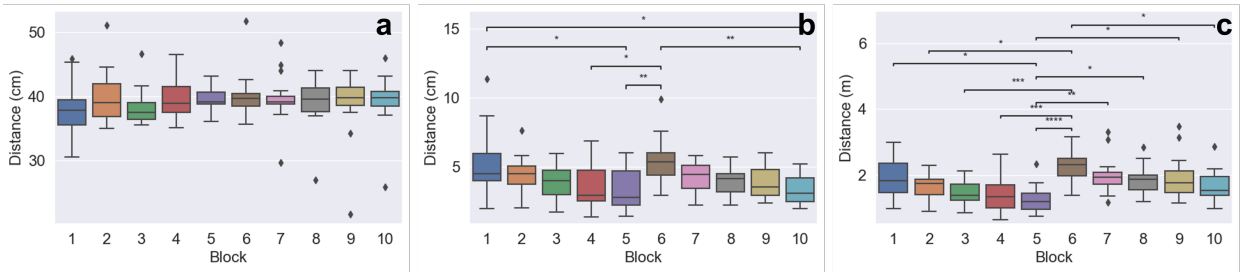


Figure 5.9: Block-wise – (a) PSD, (b) SSD, and (c) SSD_{p2p} for OM study.

Interestingly, while primary submovement distance (PSD) remained unaffected by practice over the blocks, secondary submovement distance (SSD) and point-to-point distance (SSD_{p2p}) were different. These metrics increased significantly in manipulation blocks, indicating participants were employing larger corrective movements to achieve accuracy. In addition, the participants also became better over the practice in the manipulation session, which is seen from the significant difference in the SSD value between the sixth block and the tenth block (p=0.0043) (Figure 5.9).

Moreover, the speed of these corrective movements, as measured by PV_{ss} and $AvgS_{ss}$, also increased significantly during manipulation blocks. This increment in secondary submovements showcases participants' efforts to refine their corrective strategies over time, further emphasizing the dynamic nature of human adaptation to sensory manipulations in VR (Figure 5.10).

Through this detailed block-wise analysis, I observed human resilience and adaptability, where participants

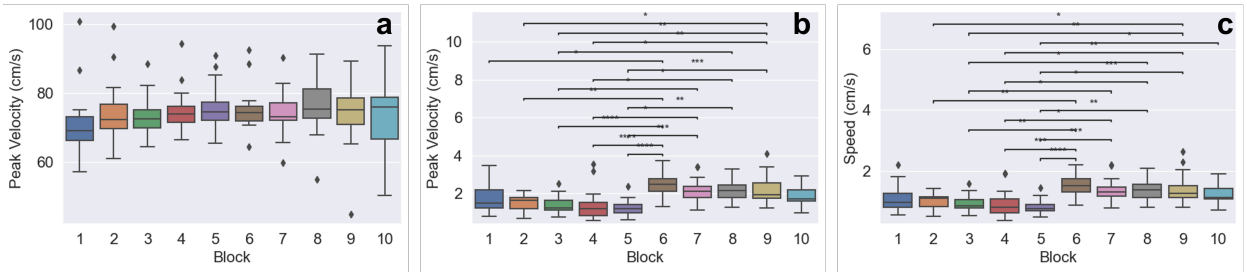


Figure 5.10: Block-wise – (a) PV, (b) PV$_{ss}$, and (c) AvgS$_{ss}$ for OM study.

not only learn to navigate the initial challenges posed by VR but also dynamically adjust their strategies to maintain performance under varied manipulations.

Impacts in Kinematic Metrics for MM

Likewise, I explore the block-wise analysis for MM, which mirrors the insights gained from the OM analysis but offers its unique perspective on the participants' experiences through perception manipulations.

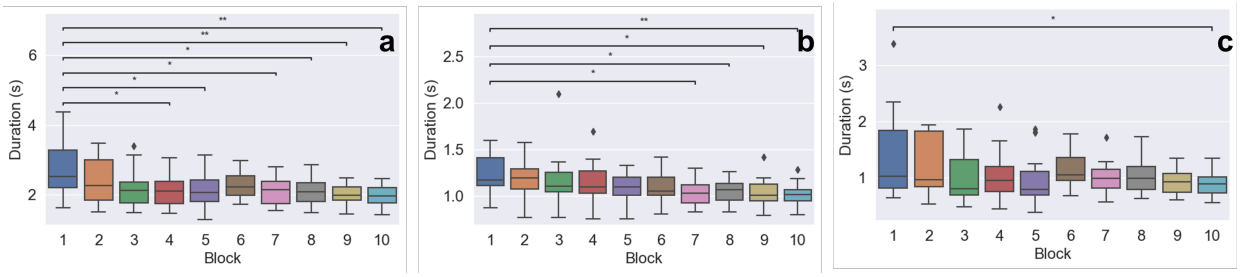


Figure 5.11: Block-wise – (a) MT, (b) T1, and (c) T2 for MM study

Similar to OM, movement time (MT) significantly differed in the early stages but converged as participants progressed through the sessions. This trend suggests an early adjustment to the VR environment, particularly evident in the primary submovement time (T1), where the initial slowness gave way to quicker, more confident movements as the sessions advanced (Figure 5.11).

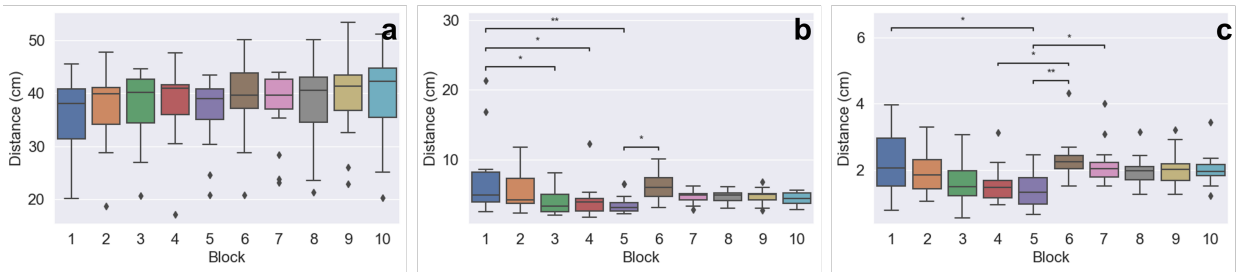


Figure 5.12: Block-wise – (a) PSD, (b) SSD, and (c) SSD$_{p2p}$ for MM study.

Upon transitioning to the manipulation phase, notable changes emerged in participants' approaches, particularly concerning their secondary submovements (T2). The introduction of magnitude manipulations prompted participants to refine their corrective movements, evidenced by an increase in secondary submovement distance (SSD) and point-to-point distance (SSD_{p2p}) during these blocks. This augmentation signifies an adjustment aimed at preserving precision amidst manipulations (Figure 5.12).

Interestingly, while the peak velocity (PV) across blocks showed minimal variance, indicating consistency in movement speed, the narratives of peak velocity during secondary submovement (PV_{ss}) and average speed ($AvgS_{ss}$) unfolded differently. Upon facing manipulations, participants not only sped up their corrective movements but also improved their efficiency, as seen in the significant uptick in PV_{ss} and $AvgS_{ss}$ speeds during later blocks. This increment suggests an adaptation of movement strategies to counterbalance the VR manipulations (Figure 5.13).

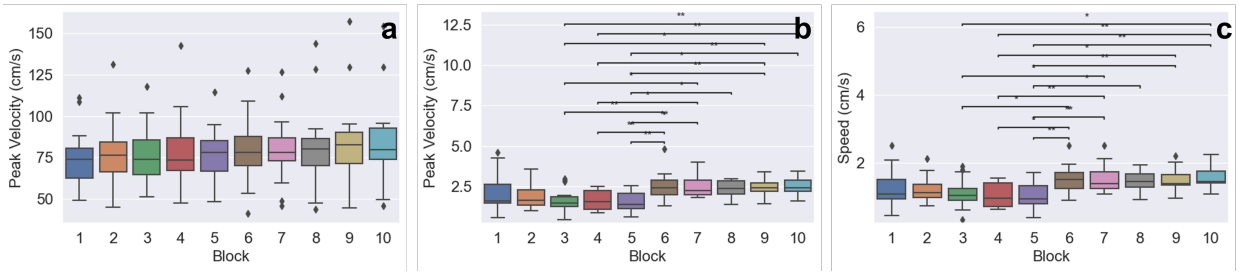


Figure 5.13: Block-wise – (a) PV, (b) PV_{ss}, and (c) $AvgS_{ss}$ for MM study.

Through this detailed exploration, the block-wise analysis for MM shows human adaptability and resilience within immersive VR environments. Participants navigated initial learning curves, adjusted to manipulations, and optimized their movement strategies to maintain performance.

5.3 Discussion

The study explored the effects of orientation and magnitude manipulations on different movement kinematics. Firstly, I present our findings regarding the impact of these manipulations on each kinematic metric. Secondly, I delve into how manipulations affect optimal motor learning strategies, comparing our results with existing literature. Lastly, I discuss the practical implications of perceptual manipulation in Human-Computer Interaction (HCI) across various applications.

5.3.1 Adaptability and Impact Across Perception Manipulations

In the level-wise analysis, MT, T1, and T2 showed differences due to manipulations, while only initial conditions BPF and BPL differ meaningfully from the baseline B. Learning effects are clear, as BPF and BPL data were gathered later. However, orientation manipulation alone did not significantly affect these metrics. In contrast, SSD and SSD_{p2p} were highly sensitive to different manipulation levels, highlighting their responsiveness to control adjustments. PV and $AvgS_{ss}$ also increased with higher manipulations, pointing to the specific impact on speed metrics in secondary movements.

In the block-wise analysis, the data confirmed the role of practice across session blocks. Metrics like MT, T1, T2, and SSD revealed significant differences in early blocks from later ones. Specifically, SSD and SSD_{p2p} showed that users adapt their control strategies over time, becoming more efficient in both types of movements. The significant differences in PV_{ss} and $AvgS_{ss}$ indicated this adaptation extends to the speed of secondary movements.

While similar differences were noticeable in both types of manipulations, the impact during scale-up (magnitude) is not prominent. This could be the reason that scaling up the movement helped the participants to reach the target earlier than other manipulations. However, discrepancies in certain kinematic metrics (e.g., PV_{ss} and $AvgS_{ss}$) were noticed during extreme scale-up manipulations, such as for a scale factor of 1.3.

5.3.2 Contrast of Perception Manipulation Effects with Prior Work

An early work by Kohli et al. [306] investigated the impact of spatial warping in virtual reality on user performance and perception. The study leveraged the concept of passive haptic feedback and redirected touching to explore how discrepancies in virtual and real object orientations affect task performance, error rates, and user detection of discrepancies. The study provided evidence that users can interact with warped virtual spaces without significant detriments to task performance or error rates compared to unwarped spaces within specific bounds of manipulation. They found that the task performance changes when users are presented with orientation manipulation beyond $\pm 18^{\circ}$. The study also suggested that users can adapt to manipulations in virtual and real object orientations. Similar to this, our study did not find any significant differences in task performances within these ranges of orientation manipulations, which aligns with the prior study.

Another study by Han et al. [307] investigated by evaluating two distinct hand remapping techniques – translational shift and interpolated reach – for enhancing hand interactions with passive haptics in VR. They presented a series of experiments and a case study and explored the efficacy, adaptability, and usability of these techniques in VR environments. In one experiment, their findings revealed that translational shift

generally outperforms interpolated reach in terms of speed and accuracy, especially in scenarios with larger mismatches between virtual and physical objects. The experiment also uncovered that offset direction (toward or away from the user) significantly affects reach time, with away offsets facilitating faster reaches. Another experiment focused on users' ability to adapt to the remapped reach techniques over time. The results showed improvement in task performance with practice, indicating that users can adjust to remapped motions. However, the degree of acclimation varied between techniques, with translational shift showing more consistent adaptation than interpolated reach. In our study, we utilized the interpolated reach approach. Similar to this study, we noticed improvement in task performance with practice.

A very recent study by Yang et al. [308] provided insightful findings on the impact of different redirected strategies (Redirected Reach (RR), Redirected Placement (RP), and a combined RR&RP method) on ballistic and corrective movements during VR interactions. The study's analysis focused on how these strategies influence the participants' movement profiles when interacting with virtual objects. The study observed that redirection strategies influence the proportion of ballistic movement in the overall task performance. Specifically, the proportion of the ballistic phase tends to decrease as the redirection offset increases, indicating that participants adjust their initial, rapid movement strategy based on the perceived alignment between the virtual and physical environments. Similarly, the study found that corrective movements became more prominent with higher redirection offsets, suggesting an increased reliance on visual feedback to align the virtual and physical hand positions. This was particularly evident when the redirection offset was beyond the detection threshold, leading to a noticeable discrepancy between virtual and physical spaces that participants needed to correct. Their findings specific to each redirected strategy:

- **RR-only**: When redirection was applied only during the "reach-to-grasp" phase, participants exhibited increased corrective movements, especially as the redirection offset increased. This indicates that participants were more sensitive to discrepancies during the grasping phase, necessitating more correction to align their movements with the virtual object.

- **RP-only**: In the RP-only condition, where redirection occurred only during the "reach-to-place" phase, a similar pattern was observed. Higher redirection offsets led to more pronounced corrective movements, highlighting the challenge participants faced in aligning the virtual and physical positions during object placement.

- **RR&RP**: The combined RR&RP method aimed to distribute the redirection offset across both the "reach-to-grasp" and "reach-to-place" phases. This strategy was found to optimize the balance between ballistic and corrective movements. Participants exhibited a more natural movement pattern with this

approach, indicating that distributing the redirection offset made the discrepancies less noticeable and reduced the need for extensive corrective movements.

Similar to Yang et al. [308], the manipulations significantly affect the secondary submovement distance, which increased with a higher degree of manipulations. This happened because the traveled distance using the ballistic movement has been decreased with the presence of higher manipulations. This phenomenon underscored the impacts of manipulations on motor performance. In contrast to Yang et al. [308], I explored other kinematic mercies, such as I also noticed that the participants sped up their movements during corrections to reach the target as quickly as possible, indicating adapting to the manipulations. For this adaptation, people primarily relied on visual feedback, indicating that proprioceptive reliance might decrease in the presence of higher degrees of manipulations. Moreover, increasing speed during secondary submovements requires additional effort and coordination from the users [230].

5.3.3 Learning Effects

The lack of significant Movement Time (MT) changes between baselines and manipulated sessions pointed to learning effects that confound the outcomes. For example, the participants improved their performance in baseline sessions over time. However, as soon as they were exposed to the manipulations in the later sessions, they could not maintain their earlier learned skills; rather, they were taking a significant amount of time to complete the tasks. This suggested a reassessment of whether the traditional comparisons between baselines and manipulations were effective for evaluating such manipulations. Future research might benefit from utilizing a counterbalanced design to control for learning effects in a mixed environment.

5.3.4 Adaptation Mechanisms

During the block-wise analysis, I observed that participants covered greater distances with higher degrees of manipulation (e.g., SSD and SSD_{p2p} metrics). Additionally, participants accelerated their movements to expedite task completion in the presence of manipulations. This behavior suggests that participants not only reacted to changes in the system (i.e., manipulations) but also adjusted to the systems over time. The incremental patterns in the kinetics metrics during manipulated sessions indicate engagement of cognitive load and motor adaptation strategies [230]. Hence, understanding the underlying factors driving these strategies is crucial for developing more efficient and less intrusive manipulations in future human-computer interaction systems.

5.3.5 Implications for Human-Computer Interaction

The observed patterns in kinematic metrics, particularly in PV_{ss} and $AvgS_{ss}$, indicating increased speed in secondary submovements, may have broader implications for user interactions with orientation-dependent interfaces or applications. These findings could inform design and user experience strategies for real-world systems involving orientation manipulations.

Magnitude manipulations seem to impose cognitive load, affecting both the speed and accuracy of interactions. This has implications for interface designs incorporating force or pressure-sensitive interactions, where high-magnitude manipulations may hinder fine-grained control, while low-magnitude manipulations may facilitate quicker interactions.

Further exploration is needed to understand the observed learning effects and adaptation mechanisms more deeply. Future studies should isolate and examine these factors closely, expanding the range of orientation manipulations examined to gain insights into scalability and practical application.

Finally, our findings offer a foundation for further research into the enduring impact of VR-induced perception manipulations on motor skills. By analyzing user responses to different manipulations, we advocate for balanced manipulation strategies in VR system design, optimizing user experience while preserving motor efficiency and minimizing cognitive demands.

5.3.6 Future Directions

Through our investigation of the impacts of manipulations on kinematic metrics, I observed phenomena of learning effects and adaptation mechanisms, which, although intriguing, remain partially explained. Future research could concentrate on isolating these variables to gain a better understanding of their individual contributions to performance. Moreover, deploying a wider range of orientation manipulations can further validate the scalability and applicability of these findings.

Chapter 6

Detecting Hand-Movements Manipulation

In Chapter 5, I explored the effects of perception manipulations on kinematic metrics during reaching tasks. These manipulations can disrupt movement strategies, impairing the efficiency of hand movements and leading to decreased task performance [306, 308]. In some cases, if employed maliciously, they may even jeopardize user safety. Additionally, dynamic manipulation techniques may inadvertently affect the precision of finely tuned motor movements, which rely on visual and kinesthetic feedback for refinement. Acquiring motor skills and the development of developing muscle memory is vital for performing tasks quickly, consistently, and stably while minimizing cognitive load [231]. However, the dynamic nature of perception manipulation, which alters the mappings between virtual and real hand movements, poses challenges to the development of such automaticity. Moreover, there are concerns that perception manipulation could be maliciously exploited to harm users, with potential adversaries including VR developers who might intentionally or unintentionally introduce harmful manipulations [35]. Instances of such exploitation have been documented, including manipulation of VR safety boundaries to control user movement without their awareness [34]. I advocate for a user's prerogative to be informed of any such potential manipulations before application usage. To this extent, recognizing the importance of user awareness regarding third-party applications that incorporate perception manipulations, I propose a method to detect them and mark them accordingly in VR stores, enhancing app usage integrity and informing user choices.

This chapter introduces *ManipulaSense*, a novel approach to detecting perception manipulations in VR, leveraging natural hand movements within the theoretical framework of Woodworth's two-component model of goal-directed movements, established in 1899 [309]. According to this model, goal-directed actions comprise

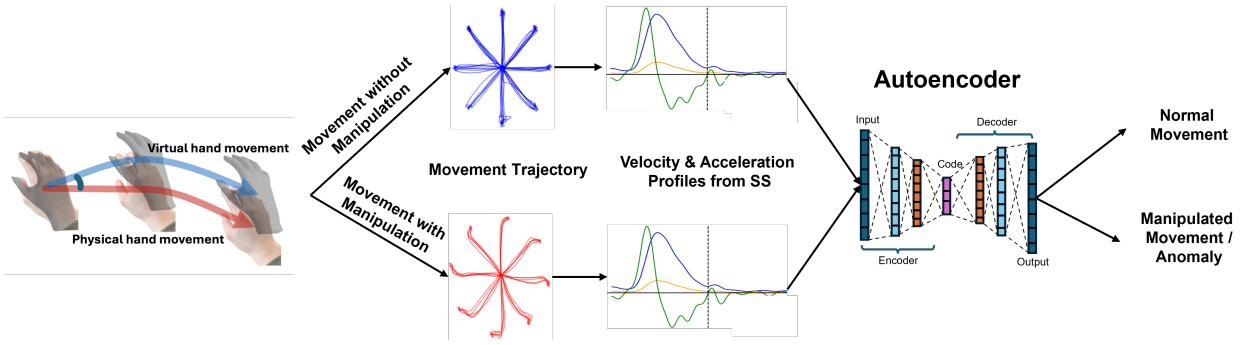


Figure 6.1: Perception Manipulation alters the user's behavior or perception during the interaction in VR. While these perception manipulations are often unnoticed by the users, their movement may be affected by the manipulation. Our work presents an Autoencoder-based method that detects the anomaly in the user's hand movement and identifies the perception manipulation.

an initial ballistic phase, directing the limb toward the target, followed by a corrective phase that adjusts for trajectory errors using visual feedback. Our observations indicate that the speed of users' corrective movements varies in response to perception manipulation, a variation not present under normal conditions. This phenomenon occurs when perception manipulation is applied in a particular direction, causing users to adjust their hand movements in the opposite direction. One can quickly notice this adjustment, which is evident in the hand movement trajectories shown in Figure 6.1 and Figure 6.5. This observation forms the basis of our hypothesis that the characteristics of hand movements during corrective phases, such as speed, acceleration, and jerk, can serve as indicators to differentiate between manipulated and normal movement patterns.

To test our hypothesis, I developed *ManipulaSense*, a semi-supervised machine learning model based on an *Autoencoder* (AE) [320, 321] for anomaly detection. ManipulaSense distinguishes between manipulated and non-manipulated instances by learning to reconstruct normal movement patterns, with the premise that anomalies—manipulated movements—will exhibit higher reconstruction errors [322, 321]. This approach leverages the assumption that normal instances are more accurately reconstructed from learned representations. At the same time, anomalies deviate significantly, resulting in higher reconstruction errors. The determination of an anomaly is contingent upon a reconstruction error exceeding a predetermined threshold. I conducted multiple evaluations of our model to optimize this threshold, selecting the optimal threshold based on the differentiation between normal and manipulated data.

The efficacy of our approach was validated through a user study involving 21 participants, where I collected hand movement data in VR under both normal (non-manipulated) and manipulated conditions. Our study design incorporated orientation manipulation at two levels ($\pm 10^{\circ}$) to simulate manipulated movement conditions. Data collection was structured into three sessions: an initial session to establish baseline hand

movements (S1), a second session to collect data under manipulated conditions (S2), and a final session to gather additional baseline data (S3). The model was trained in a semi-supervised manner using only S1 data to learn regular hand movement patterns, with S2 and S3 data serving to evaluate the model's ability to accurately detect manipulated instances as anomalies and baseline instances as normal. Our findings, derived from a within-subject analysis employing a 70-30 train-test split, demonstrate that our model achieves an anomaly detection accuracy of 97.7% (F1-score: 98.1%). Further validation through leave-one-participant-out (LOPO) cross-validation confirmed the robustness of our model in detecting anomalies across new users, with an accuracy of 93.7% (F1-score: 93.9%).

6.1 ManipulaSense

Our methodology for detecting perception manipulation through user hand movements is inspired by the two-component model of rapid aimed limb movements, as described by Woodworth [309]. This model depicts hand movements into *primary submovements* (PS) for the initial target approach and *secondary submovements* (SS) for fine-tuning the position. The initial, primary submovement is believed to be ballistic and almost entirely preprogrammed and serves to propel the limb most of the distance between the starting position and the target. If the PS does not land in the target region, then a *corrective* SS may also be produced [319, 230]. I observed that manipulations in perception, such as changes in orientation, distinctly affect the speed of these corrective movements. This variation and differences in acceleration form the basis of our hypothesis: the dynamics of corrective movements, particularly their speed and acceleration, are indicative of manipulated versus normal movements.

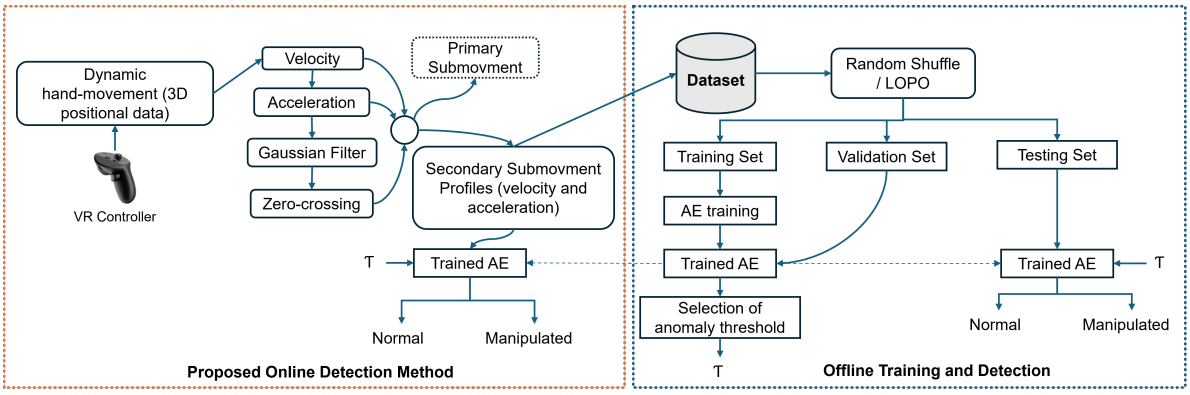


Figure 6.2: ManipulaSense overview.

Our methodology for detecting anomalies using VR hand movements follows a structured process. First, I compute the speed as the initial temporal derivative of the 3D hand movements captured from the VR

controller and then calculate the acceleration as the speed's derivative. I then smooth the speed and acceleration profiles using a Gaussian filter to ensure a cleaner analysis. The identification of the SS is crucial to our approach. Similar to Abrams & Pratt's method [319], I identify the start of SS at the second zero-crossing in the acceleration profile, which occurs right after the peak speed. This zero-crossing signifies a transition from deceleration to acceleration, marking the beginning of corrective movements. I then segment the raw speed and acceleration data based on the identified SS start point. Finally, I utilize the raw speed and acceleration data, specifically from the SS segment, as input features for our model training.

Leveraging these segmented corrective movements, I employ anomaly detection via an Autoencoder – a deep learning model renowned for its capacity to identify abnormal patterns by learning to compress and reconstruct normal data with minimal error. An LSTM-based Autoencoder is particularly suited for this task due to its proficiency in handling sequential data. By training exclusively on data representing unmanipulated hand movements, the Autoencoder develops a normative model of such movements. Consequently, manipulated movements produce a notable deviation in reconstruction, showing increased reconstruction loss, which I use to identify manipulations.

After training, I establish a threshold for anomaly detection by analyzing reconstruction errors, enabling the differentiation of manipulated movements. This threshold, determined through offline analysis, is then applied in a real-time system to identify manipulation as it occurs. The methodology – from offline analysis to real-time application – is illustrated in Figure 6.2. Now, I discuss the LSTM Autoencoder and threshold selection process in detail.

6.1.1 LSTM Autoencoder (AE) Architecture

Our LSTM AE model comprises two main components: an encoder and a decoder. The encoder compresses the input sequence of hand movement data into a lower-dimensional latent space, while the decoder aims to reconstruct the original sequence from this compressed representation.

- **Encoder:** It transforms the input sequence (x) into a compact latent representation (z). The encoding function can be represented mathematically as $z = f(x)$, where f is a nonlinear transformation learned during training. The encoder consists of LSTM layers, starting with an input dimension of 2 (representing kinematic features such as speed and acceleration) and culminating in an encoding dimension of 7. Intermediate hidden dimensions are set to 64 (Figure 6.3).

- **Decoder:** It attempts to reconstruct the input (x) sequence from the compact latent representation (z). The reconstruction $\hat{x}$ is obtained by applying a reverse transformation g to the latent representation: $\hat{x} = g(z)$. The goal is to make $\hat{x}$ as close to x as possible, reflecting a minimal loss of information during

encoding. The decoder's architecture mirrors the encoder, utilizing LSTM layers to expand from the encoding dimension back to the original feature space (Figure 6.3).

The composition of these two functions can describe the entire AE: $\hat{x} = g(f(x))$. The reconstruction error (loss) quantifies the difference between the original input x and its reconstruction $\hat{x}$. For our model, the Mean Absolute Error (L1 loss) is used as the loss function, which is defined for each data point as: $L(x,\ \hat{x}) = \sum_{i=1}^{n} |x_i - \hat{x}_i|$, where n is the dimensionality of the input data. The objective during training is to minimize this loss across all data points in the training set, thereby improving the AE's ability to accurately reconstruct the input data from its latent representation.

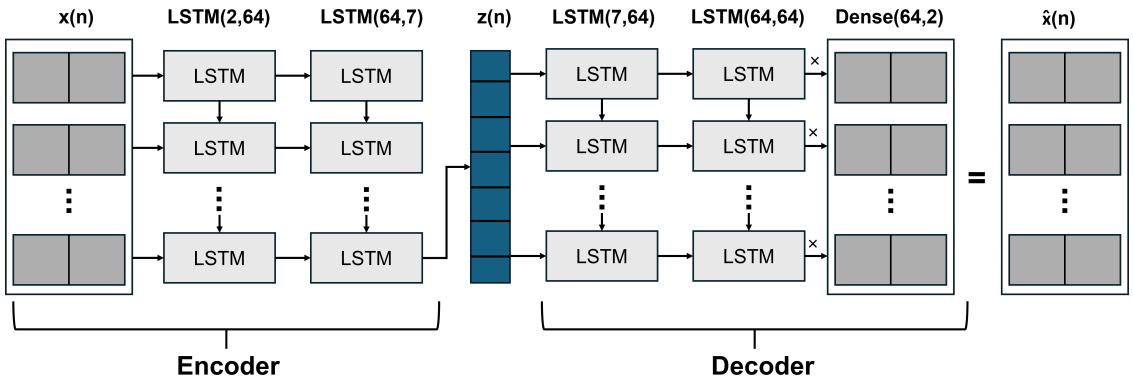


Figure 6.3: AutoEncoder (AE) Architecture.

In anomaly detection, the reconstruction error serves as a critical metric. Since the AE is trained only on normal (non-manipulated) data, it learns to reconstruct these data points accurately, resulting in a low reconstruction error for similar data. However, for anomalous (manipulated) data points, the model's reconstruction will be poor, leading to a higher reconstruction error. To detect anomalies, a threshold τ is set based on the distribution of reconstruction errors on a validation set comprising normal data. An input x is considered anomalous if $L(x, \hat{x}) > \tau$.

6.1.2 Threshold Selection

Our methodology for detecting perception manipulation leverages an AE trained exclusively on normal data. This approach is predicated on the principle that such an AE will accurately reconstruct normal data, indicated by a low reconstruction error, while failing to do so for manipulated data results in a high reconstruction error. The key to detecting manipulation lies in setting an appropriate anomaly threshold based on the reconstruction error (RE). A test sample exceeding this threshold in RE is classified as an anomaly. In contrast, samples below the threshold are considered normal. The selection of this threshold is critical; an inadequately set threshold could lead to a significant number of false detections.

In this chapter, I proposed ManipulaSense, a perception manipulation detection model that employs a stochastic approach to select the anomaly threshold, which is crucial for distinguishing between normal and manipulated samples. Our methodology is rooted in the balance between maximizing the True Positive Rate (TPR) and minimizing the False Positive Rate (FPR), using the Receiver Operating Characteristic (ROC) curve [323] to find the optimal threshold, τ. The formal equation for determining τ is:

$$\tau = \underset{\tau=l_{min}}{\overset{\tau=l_{max}}{\operatorname{argmax}}}(\lambda \times TPR(\tau) - (1-\lambda) \times FPR(\tau)) \tag{6.1}$$

Here, τ is the threshold, $TPR(\tau)$ is the True Positive Rate at threshold τ, $FPR(\tau)$ is the False Positive Rate at threshold τ, and λ is a weighting factor that balances the importance of maximizing TPR against minimizing FPR. Moreover, l_{min} and l_{max} represent the minimum and maximum reconstruction error (loss) from the training dataset. The choice of λ depends on the application's tolerance for false positives versus false negatives. A λ closer to 1 emphasizes maximizing TPR, whereas a λ closer to 0 emphasizes minimizing FPR.

To integrate this approach within ManipulaSense, I focus on the significance of the threshold value by analyzing the distribution of reconstruction loss in our dataset. During the training phase, our model is exposed solely to normal samples. In contrast, normal and manipulated samples populate the validation and testing datasets. The threshold τ is calculated based on the reconstruction errors observed in these datasets. I investigate the range of reconstruction errors (minimum to maximum) derived from the training set at intervals of 0.2 to ascertain the most effective threshold. This process ensures our model's ability to accurately detect manipulated movements, leveraging a stochastic method for threshold selection that reflects the nuanced interplay between maximizing detection accuracy and minimizing false positives. In our methodology, I found the detection threshold for perception manipulation to be 6, which equals $\mu(train\_loss) + 3.4 \times \sigma(train\_loss)$. Here, $\mu(train\_loss)$ represents the average loss incurred during training, while $\sigma(train\_loss)$ denotes the variability or standard deviation of the training loss. This threshold is consistent across new users and effectively works for hand movement data (Section 6.2.5).

6.2 Experiments

6.2.1 Dataset

In the absence of publicly available datasets, I conducted a user study to gather manipulated and non-manipulated hand movement data during reaching tasks within a VR environment. Participants were instructed to execute a task that involved grabbing an object, moving it, and placing it in a specified target

location, thereby incorporating both target selection and point-to-point reaching elements. To collect data on manipulated hand movements, I employed *angle redirection manipulation*, also known as Orientation Manipulation. Specifically, I implemented two degrees of orientation manipulation, $\pm 10^{\circ}$, based on insights from previous studies [303, 302, 304]. These angles were selected to ensure the manipulation remained imperceptible to the participants. The study protocol was approved by the Institutional Review Board (IRB) – UVA.

Participants

A total of 21 participants were recruited for the study. All participants were right-handed ($M = 16$, $F = 5$). The group's average age was 21.8 years ($\sigma = 4.8$). Among the participants, 17 had previously used VR for gaming and academic research, one routinely used VR in their daily life, and three had no prior exposure to VR. Participants were recruited through mailing lists, Slack channels of student groups, and word-of-mouth referrals. Each participant received a $20 gift card as compensation for their time.

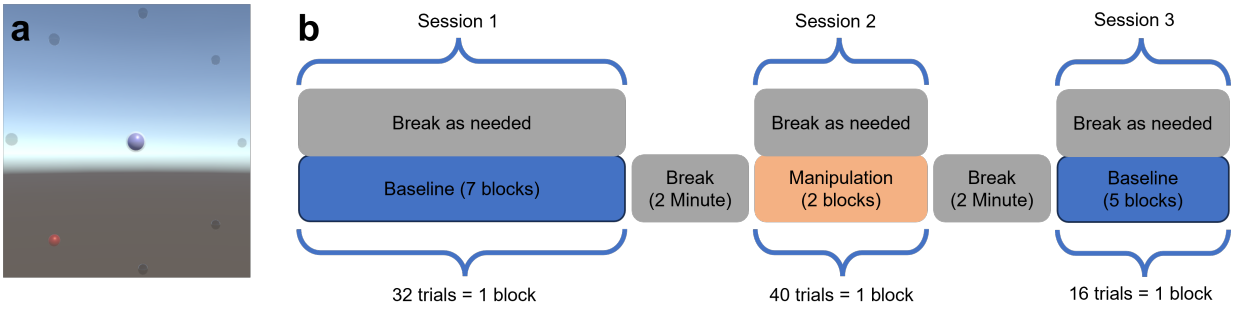


Figure 6.4: (a) Screenshot of the task scene. (b) The study sessions are shown as a sequence of blocks in a diagram.

Study Design and Procedure

The study employed a within-subjects design and was conducted in a controlled lab environment. All participants were provided with an *Meta Quest Pro* VR headset and *Controller*. The participants sat in a stationary chair that was secured to the floor. The participants were briefed on the study's purpose and possible manipulation within some trials. However, they were not informed about which specific trials included these manipulations.

In the study, the participants were asked to complete a series of reaching tasks in VR. In the study environment, the blue sphere was positioned at the shoulder level of the participant, approximately 25 cm away from the shoulder. The red target sphere was placed 35 cm away from the blue sphere in one of the eight directions (See Figure 6.4a). In each task, participants were asked to grab a blue sphere at the center of their

view and place it inside the red target sphere as quickly and accurately as possible. If the blue sphere was not correctly placed inside the red sphere, the trial was considered incomplete, and the participant was asked to try it again. The participants were instructed to rest as needed between trials to minimize any effects of fatigue. Spatial parameters were standardized in Unity units, with object sizes and distances predefined to maintain uniformity across sessions. The experimenter calibrated the VR setup at each session's onset to ensure consistent target locations for all participants. Once the participants became familiar with the setup and study environment, the experimenter started the actual data collection session.

The study was divided into three sessions (Figure 6.4b) to gather baseline and manipulated hand movement data separately. The initial session (S1) has the largest number of trials (224 trials) and aims to establish a baseline by allowing participants to familiarize themselves with the VR environment and refine their hand movements to reach an optimal movement strategy.

The second session (S2) introduced manipulations to observe how participants adjusted their strategies in response. The final session (S3) revisited baseline conditions to assess if participants could return to their original movement strategies after experiencing manipulations, providing a test set for our model.

The S1 consisted of seven blocks that included 32 trials across eight different target orientations (trials per participant $= 7 \times 32$). The S2 focused on collecting manipulated data through two blocks of 40 trials each (trials per participant $= 2 \times 40$). I randomized the two orientation manipulations such as $\pm 10^{\circ}$ during S2. In S3, I again collected baseline trials, structuring five blocks containing 16 trials each (trials per participant $= 5 \times 16$). During all the sessions, the target sequence was randomized to ensure unbiased results, with breaks interspersed to alleviate fatigue. Manipulations were confined to the XY plane to account for participants' depth perception sensitivity [305]. The manipulative adjustments were calculated using designated equations tailored to orientation manipulation conditions (Equation 6.2).

$$\left.\begin{aligned}
\Delta X &= current_hand_position_X - previous_hand_position_X \\
\Delta Y &= current_hand_position_Y - previous_hand_position_Y \\
\Delta Z &= current_hand_position_Z - previous_hand_position_Z \\
\Delta X_Angle &= \Delta X \times \cos\theta - \Delta Y \times \sin\theta \\
\Delta Y_Angle &= \Delta X \times \sin\theta + \Delta Y \times \cos\theta \\
current_hand_position &= Vector(current_hand_position_X + \Delta X_Angle, \\
&\qquad current_hand_position_Y + \Delta Y_Angle, \\
&\qquad current_hand_position_Z + \Delta Z)
\end{aligned}\right\} \tag{6.2}$$

In Equation 6.2, ΔX, ΔY, and ΔZ represent the change in hand position along the X, Y, and Z axes, respectively, calculated as the difference between the current and previous hand positions. The angles ΔX_Angle and ΔY_Angle are derived from the original changes in position (ΔX, ΔY) adjusted by the rotation angle θ, facilitating the manipulation of hand movement in the VR space. The updated current hand position is then recalculated by applying these angular adjustments to the original positions, effectively simulating the hand's movement within the virtual environment. This approach allows for a nuanced control and manipulation of hand movements in VR. The study yielded 4,704 baseline, 1,680 manipulation, and 1,680 baseline trials from sessions S1, S2, and S3, respectively.

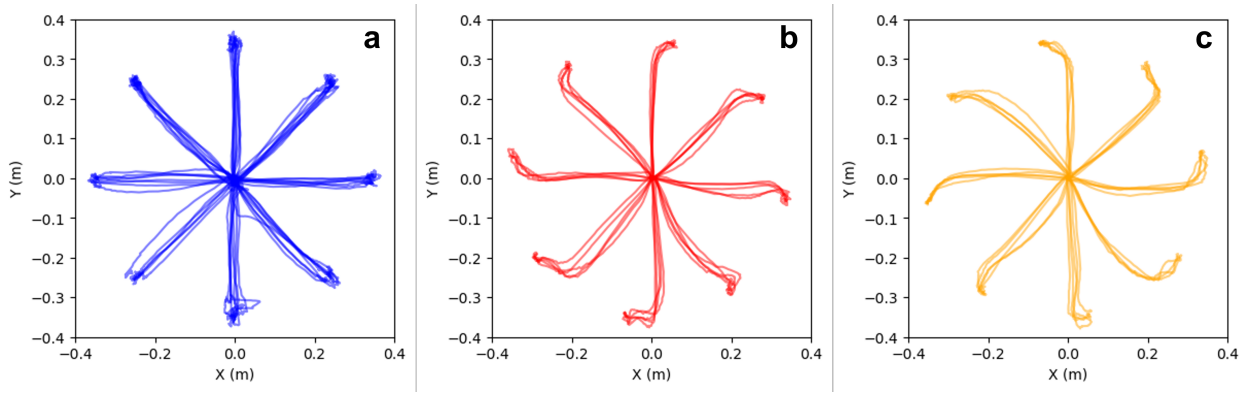


Figure 6.5: Hand Movement Trajectories (True Controller's Position) – (a) Under Normal Conditions, (b) Experiencing -10° Manipulations, and (c) Subject to $+10^{\circ}$ Manipulations.

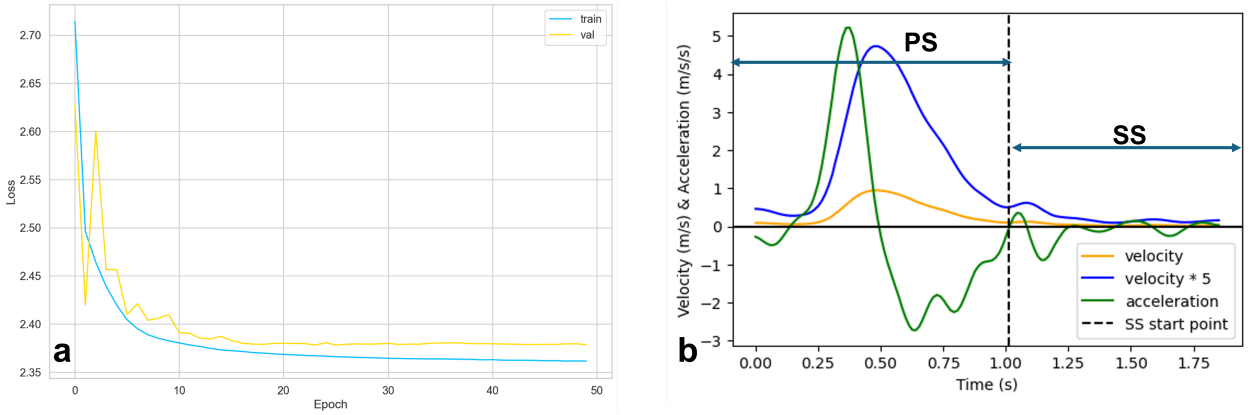


Figure 6.6: (a) Training and evaluation loss trends across epochs ranging from 1 to 50. (b) It depicts velocity and acceleration profiles for a single hand movement, showing actual velocity and acceleration alongside velocity scaled by a factor of 5 for clearer visualization. This part also marks the segmentation between PSs and SSs, with the initiation of the SS indicated by a vertical dashed line.

6.2.2 Data preprocessing

The experimental software recorded the virtual controller's position in X, Y, and Z axes at a 72-Hz sampling rate, matching the VR system's refresh rate. It captured the true controller position during both standard and manipulation phases, displaying an altered controller position to participants during manipulations. A few hand movement samples showing their trajectories are illustrated in Figure 6.5. The software logged each trial's start and end times – marking the grabbing and placing of the blue sphere, respectively – excluding trials where the sphere was misplaced. Positional data between start and end points were segmented and interpolated for consistency at 72 Hz. This processed data facilitated the calculation of cumulative Euclidean distances, velocities, and accelerations per trial, with velocity and acceleration profiles smoothed using Gaussian kernel filters with standard deviations of 5 and 4, respectively [318]. These profiles helped identify PSs and SSs, adopting a method from [319, 230]. The segmentation between PS and SS is illustrated in Figure 6.6(b). After identifying the SS portion, I extracted the time-series speed and acceleration profiles as features for training the AE model. Since the SS portion from the users' data has a dynamic length, I made a uniform length of 72 time-series samples, which is a 1-s (sampling rate: 72 Hz) window. Therefore, the time-series input shape for our model is $(2,\ 72)$, with a window size of 72 and a dimension of two features (speed and acceleration). To make the input uniform length, I resampled the SS portion using *scipy* [324].

To standardize the feature scales within our dataset, I applied Min-Max Normalization, rescaling each feature to a $[0, 1]$ range. This normalization was executed using the Equation 6.3:

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{6.3}$$

Here, x_i denotes the d-dimensional feature vector from the training dataset, and z_i represents the normalized data for the i-th instance. I determined $\min(x)$ and $\max(x)$ using the training dataset (S1) and applied these values consistently to normalize subsequent datasets (S2 and S3) for uniformity across the data processing pipeline.

6.2.3 Evaluation Metrics

I used accuracy and F1-score to evaluate the performance of our proposed method. In addition, I used *True Positive Rate* (TPR), aka *Recall* and *False Positive Rate* (FPR), to determine the anomaly detection threshold, τ. These metrics are calculated using four measures: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). A TP is when a manipulated hand movement instance is correctly labeled and measured as TP. Similarly, a TN is when a normal hand movement instance is labeled as normal and measured as TN. On the other hand, an FP is when a normal instance is labeled as a manipulated

instance and measured as an FP. Similarly, an FN is when a manipulated instance is labeled as normal and measured as FN. Based on these metrics, accuracy, F1-score, TPR, and FPR are calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{6.4}$$

$$F1\text{-}score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{6.5}$$

$$Precision = \frac{TP}{TP + FP} \tag{6.6}$$

$$Recall \text{ or } TPR = \frac{TP}{TP + FN} \tag{6.7}$$

$$FPR = \frac{FP}{FP + TN} \tag{6.8}$$

6.2.4 Experimental Setup

The training involves optimizing the AE to minimize the reconstruction error between the input and output sequences. I employ the *Adam optimizer* with a learning rate of 1×10^{-3}, leveraging its adaptive learning rate properties to efficiently converge to optimal weights. The loss function used is the *Mean Absolute Error* (L1 Loss), selected for its robustness to outliers, which sums the absolute differences between predicted and true values, facilitating a direct measure of reconstruction fidelity. The input dimension is 2×72, where 2 represents the kinematic features such as speed and acceleration, and 72 is the time-series sequence length. The intermediate encoding and hidden dimensions (for both the encoder and decoder) are 7 and 64, respectively. The model was trained for 50 epochs for both analyses discussed below.

In our study, I implemented two distinct analytical approaches: 1) a within-subjects analysis, also known as population analysis, and 2) a leave-one-participant-out (LOPO) cross-validation, referred to as between-subjects analysis. The choice to conduct a within-subjects analysis was driven by multiple factors. Primarily, I sought to explore the effectiveness of our model in personalized settings, especially in scenarios where it would be deployed online. This analysis framework allowed us to closely monitor the model's capability to identify anomalies individually, thereby assessing its accuracy and robustness across varied personal user data.

To facilitate the within-subjects analysis, I merged all participants' data from the first session (S1). I then proceeded to partition this data, allocating 70% for training the AE model, adhering to a 70-30 train-test split ratio. The residual 30% of data was further divided equally into two segments (50-50 evaluation-test split), with one half reserved for validation during the model's training phase and the other half set aside for final evaluation, alongside data from sessions two (S2) and three (S3). The trend between training and evaluation loss over epochs during within-subjects training is reported in Figure 6.6(a).

Given the inherent variability in users' hand movement data, attributed to distinct movement styles, unique physiological characteristics, and varying speeds of hand motion, it was imperative to evaluate our model's adaptability to new users. The LOPO cross-validation method was employed to address this aspect. This involved training the model on the S1 data from twenty participants and testing it on the data from the remaining participants alongside the S2 and S3 datasets. This process was repeated 21 times, corresponding to the total number of participants in our study, ensuring that each participant's data was used as a test set once. The cumulative findings were synthesized by calculating the average of all iterations.

ManipulaSense was implemented using the *PyTorch* deep learning framework. The experimental setup was conducted in *JupyterLab*, hosted on a server cluster equipped with an NVIDIA A100 GPU, 256 GB of RAM, and an 8-core CPU.

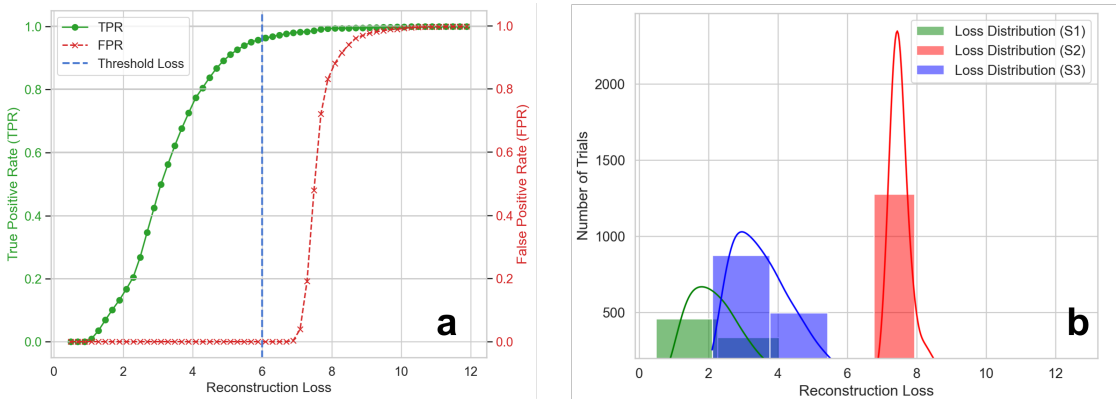


Figure 6.7: Within-subjects analysis – (a) shows the TPR and FPR across reconstruction losses ranging from 0 to 12, with the optimal threshold marked at 6, and (b) illustrates the reconstruction loss distribution for data across three sessions.

6.2.5 Experimental Results

Within-subjects Results

In the within-subjects analysis, our AE model achieved an accuracy of 97.7% and an F1-score of 98.1%. The TPR and FPR across varying reconstruction losses from 2 to 12 are illustrated in Figure 6.7(a). At the

anomaly threshold of 6 – marked by a vertical line in the figure – our model recorded a TPR of 96.3% and an FPR of 0.0%. Figure 6.7(b) presents the distribution of reconstruction loss for data from all three sessions, supporting the selection of 6 as the optimal threshold. This distribution substantiates the model's ability to differentiate between normal and anomalous instances, justifying the threshold choice. Moreover, I report the ROC curve for this analysis in Figure 6.8(a).

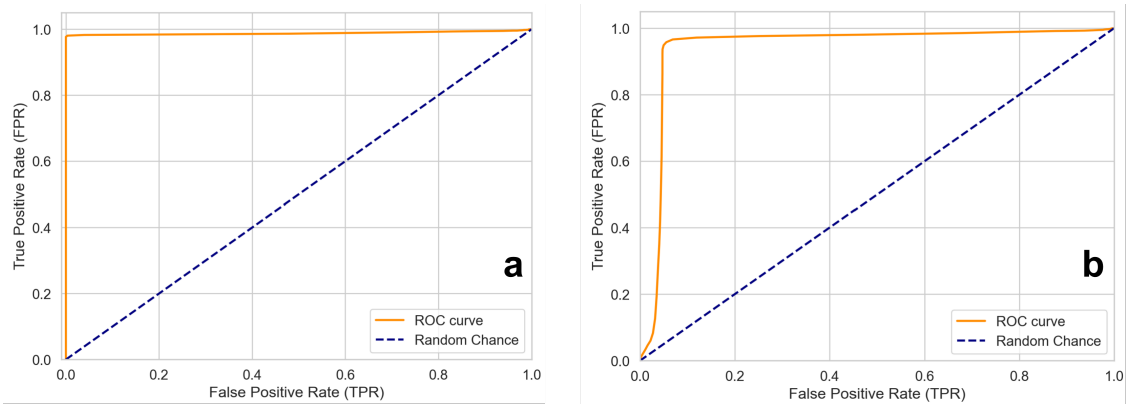


Figure 6.8: ROC curves for – (a) within-subjects analysis and (b) LOPO analysis.

LOPO Results

In the between-subjects analysis, our AE model demonstrated an average accuracy of 93.7% and an average F1-score of 93.9%. The TPR and FPR for reconstruction losses ranging from 2 to 12 are depicted in Figure 6.9(a). At the established anomaly threshold of 6 – indicated by a vertical line within the figure – the model achieved an average TPR of 92.4% and an average FPR of 4.7%. Figure 6.9(b) shows the reconstruction loss distribution (aggregated from 21 models) for the dataset spanning all three sessions, reinforcing 6 as the optimal threshold. This distribution confirms the model's efficacy in distinguishing normal from anomalous instances, validating our choice of threshold. It is important to note that the same threshold was found to be optimal in both our within-subjects and between-subjects analyses, illustrating the model's consistent performance in maximizing TPR and minimizing FPR. Additionally, the ROC curve for this analysis is reported in Figure 6.8(b).

6.3 Discussion

In this chapter, I demonstrate that users' hand movement behavior is influenced by the presence of orientation perception manipulation, even when the manipulation degree is subtle enough to go unnoticed by the users. Through the development and validation of an AE-based algorithm, I further show that variations in hand

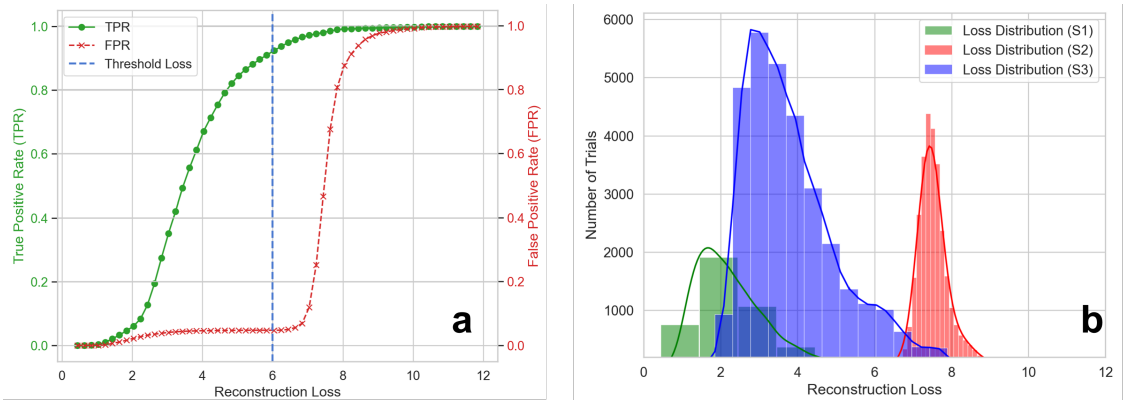


Figure 6.9: LOPO analysis – (a) presents the TPR and FPR over reconstruction loss values from 0 to 12, pinpointing the optimal threshold at 6, and (b) depicts the distribution of reconstruction loss across three session datasets in the LOPO evaluation, showcasing aggregated frequencies from all 21 models.

movement can effectively detect the presence of perception manipulation. Our experimental results indicate that the behavioral changes induced by perception manipulation are significant across participants, and using a heuristically determined threshold enabled the classification of manipulated interaction samples with high accuracy (93.7%). It is important to note that this high accuracy was achieved using only interaction samples without perception manipulation. In actual scenarios, the system will not know the presence of perception manipulation, making it impossible to train the detection model with manipulated interaction samples. Our anomaly detection approach will enable the automatic detection of manipulation based on the user's behavior, provided that the user has engaged in a sufficient number of interactions with the VR system.

Here, I discuss the considerations for implementing this approach in a VR system and the limitations of ManipulaSense.

6.3.1 Applicability of Anomaly Threshold in Online Settings

The method I used to set the anomaly threshold (τ), relying on TPR and FPR, is based on having both normal and manipulated data available, which is the case in offline experiments. However, in real-world situations, access to both data types is often impossible.

A promising aspect is that, from our analysis, the anomaly threshold determined was consistent across both within-subject and LOPO analyses. This consistency across different analyses indicates that the threshold can effectively differentiate between types of hand movements, even with new users in the LOPO analysis. Therefore, I believe this threshold can be applied to online settings. However, a larger-scale study is needed to validate it. If there are significant differences in the anomaly threshold across different people in the larger-scale study, I may employ adaptive thresholding based on the distribution of the user's normal data.

6.3.2 Selection of Velocity and Acceleration as Features for Model Training

In our research, velocity and acceleration profiles of secondary submovements were chosen as key features for anomaly detection. This decision stems from the observable significant differences in these profiles between normal and manipulated hand movements during tasks. Unlike higher-order derivatives such as Jerk (the rate of change of acceleration), velocity and acceleration provided distinguishable patterns that were critical for effective anomaly detection.

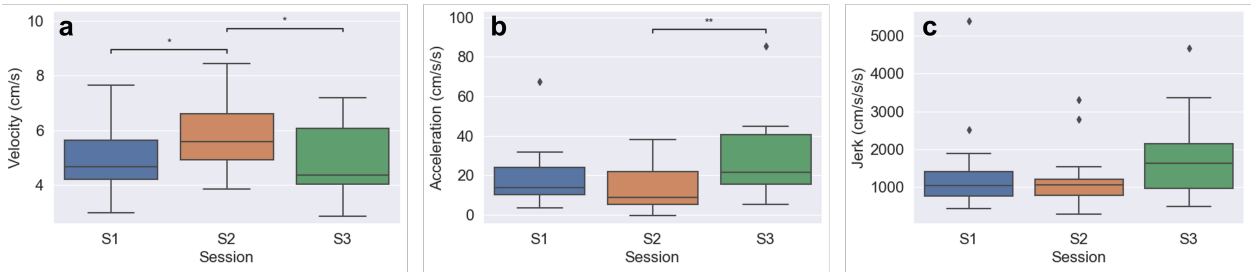


Figure 6.10: Secondary Submovement Kinematic Metrics – (a) Mean Velocity, (b) Mean Acceleration, and (c) Mean Jerk across data collected over different sessions.

The impact of these kinematic metrics was tested using a one-way repeated-measures ANOVA, with any significant differences across multiple means further analyzed using Tukey's HSD post hoc comparisons ($p < 0.05$). The analysis did not yield significant distinctions in the Jerk profile, leading to its exclusion from the model training process. The detailed statistical outcomes supporting this decision are presented in Figure 6.10, underscoring the rationale behind focusing on velocity and acceleration as primary features for detecting anomalies in hand movement within virtual environments. Additionally, I want to emphasize not including PS as a feature since the ANOVA test did not find an effect on kinematic metrics in the case of manipulations.

6.3.3 Potential Integration with VR Systems

Our research presents a viable approach for the real-time detection of manipulated movements within VR environments, aimed at bolstering user safety through early identification of potential manipulations. Although our initial analysis was performed offline, I envision two distinct deployment strategies aligning with our within-subject and LOPO analyses, each with its accuracy and user convenience trade-offs.

The first strategy involves a preparatory phase where a session is dedicated to collecting hand movement data from each user upon their first interaction with the VR application. This personalized data collection aims to fine-tune the anomaly detection model to the user's specific movement patterns, mirroring the within-subjects analysis approach. While this method promises higher accuracy (e.g., 97.7%) in identifying

manipulations by tailoring the detection to individual behaviors, it requires users to undergo an initial data collection process, potentially delaying their access to the VR environment.

Alternatively, the second strategy bypasses the data collection session, deploying the model based on a generalized dataset akin to our LOPO analysis. This approach allows users immediate access to the VR system without the need for preliminary data gathering. However, while convenient, it may not achieve the same level of personalized accuracy (rather 93.7%) as the first method, given the model's reliance on a broader dataset to identify anomalies.

Regardless of the deployment method, upon detecting manipulative behaviors in any third-party VR application, our system would proactively issue warnings within the VR app store, alerting users to potential risks before they engage with the content. This mechanism is vital for ensuring users are informed about the applications they choose to interact with, particularly when it comes to distinguishing between harmless entertainment apps and those that potentially compromise their personal data or pose a physical threat. By providing users with preemptive warnings, the system would empower them to make informed decisions regarding application usage, thereby mitigating the risk of exposure to malicious software.

Our proposed integration of anomaly detection into VR systems offers a strategic choice between personalized accuracy and immediate deployment. By considering the trade-offs between these approaches, VR platforms can adopt a model that best suits their operational requirements and user safety priorities, marking a significant step forward in enhancing the security and user experience of VR technologies.

6.3.4 Adopting a Semi-Supervised Approach for Personalized Anomaly Detection

Given the personalized nature of VR experiences, detecting manipulations requires an approach that can adapt to individual user behaviors. Recognizing that each user has unique hand movement patterns, a tailored anomaly detection model becomes essential for ensuring robust and personalized protection. The challenge, however, lies in the difficulty of acquiring manipulated movement data in real time within a VR environment, as collecting normal hand movement data is comparatively straightforward.

To address this, I have chosen to implement a semi-supervised learning approach. This strategy allows the model to initially learn from a substantial amount of easily obtainable normal behavior data. Subsequently, it can incrementally adapt to each new user by integrating a small subset of their data, thereby enhancing its ability to discern between normal and abnormal (manipulated) movements. This method facilitates the continuous refinement of the model, ensuring its effectiveness across a diverse user base without the necessity for explicit examples of manipulated movements.

The challenge of deploying manipulation detection in the practical scenario is to fine-tune the model for personalized settings. This is because I cannot differentiate between normal and manipulated movements in real-world scenarios, making it difficult to fine-tune the model using supervised learning. However, I can overcome this challenge by leveraging semi-supervised learning, which takes advantage of the abundance of normal movement data in real-world situations since I can collect normal data by having controlled sessions where I can ensure that there is no manipulation. Once the model is fine-tuned, the same anomaly detection threshold can be applied to online settings, as demonstrated by our analysis. Our methodology not only enhances the accuracy of anomaly detection but also ensures that the VR system remains user-centric, providing personalized protection against manipulative behaviors.

6.3.5 Limitations and Future Work

- Our study presents a novel approach for detecting manipulation within VR environments, yet it is not without its limitations. A primary constraint stems from our reliance on analyzing SSs in users' hand movements. Notably, not every hand movement includes an SS; certain movements conclude with just the PS, especially when users efficiently reach their target, aligning with the principles of optimality in movement [230]. In our dataset segmentation, this phenomenon was evident, with a proportion of movements across sessions S1, S2, and S3, concluding with PM alone. Specifically, 8.2%, 10.12%, and 10.29% of movements in sessions S1, S2, and S3, respectively, lacked SS. Despite this, our analysis supports the premise that a significant majority ($\approx 90\%$) of hand movements do encompass SS, suggesting that even with habitual VR use, sufficient data for manipulation detection remains accessible.

- Another limitation pertains to the duration of our study. The research was conducted over a single hour-long session, capturing normal and manipulated hand movements within this timeframe. The impact of extended VR usage on user behavior, potentially spanning multiple days, remains unexplored. Changes in user behavior over longer periods could influence the study's outcomes, necessitating further exploration to understand these dynamics fully.

- Additionally, our investigation focused exclusively on orientation manipulation, specifically at $\pm 10^{\circ}$ levels, without considering other forms of manipulation prevalent in VR settings, such as gain manipulations [217, 220].

- Some perception manipulation techniques, such as redirected walking [325, 224, 226, 326], do not involve hand movement. A future study will need to investigate if similar approaches can be used for detecting different types of perception manipulations.

- The threshold I used to build ManipulaSense proved effective for orientation manipulation detection, its applicability to other forms of manipulations, such as those involving magnitude changes, might require additional study to identify a suitable threshold for those scenarios. This aspect highlights the need for further research to extend the threshold's utility across various manipulation types in VR environments.

Chapter 7

Summary and Future Work

This thesis focuses on making gestures more usable so that users can adopt gesture interactions for micro-interactions on an everyday basis. I proposed SequenceSense, which helped the designers build more conflict-free user gestures. Essentially, this tool helps general users indirectly since the designers are responsible for gesture design. If the designed gestures are not good enough, the general users will quit using the system in the long run. Therefore, ensuring an efficient gesture design process is important so that the designers can easily design conflict-free gestures with minimal effort. During the gesture design process, the gesture designers consider the gestures to be comfortable, memorable, and socially acceptable. However, they struggle to make those gestures free of conflict. Therefore, they often develop complex gesture sets that might be uncomfortable to perform; moreover, general users find those gestures hard to remember. This is where SequenceSense would benefit the designers in the process of designing more conflict-free gestures by considering other properties of gestures, such as comfortability, memorability, and social acceptability. In a user study with gesture designers, I found that using SequenceSense, the designers were able to design easy-to-remember, shorter-length gestures that are also conflict-free. In this way, SequenceSense covers a broader range of good gesture properties, as shown in Figure 7.1.

On the other hand, the UnifiedSense system directly helps general users reuse their gestures even when the original gesture-sensing device is missing. It facilitates users' reuse of their muscle memory by unifying interactions across devices. While developing UnifiedSense, I integrated a gesture detector that rejects false activations to ensure the gestures are usable in the wild. In this way, UnifiedSense contributes to the design of conflict-free gestures. Moreover, the gesture sets I targeted for implementing UnifiedSense are chosen from existing gestures that people use these days. They are easy to perform, remember, and socially acceptable. How UnifiedSense contributes to achieving good gesture properties for usable gesture interactions is shown in

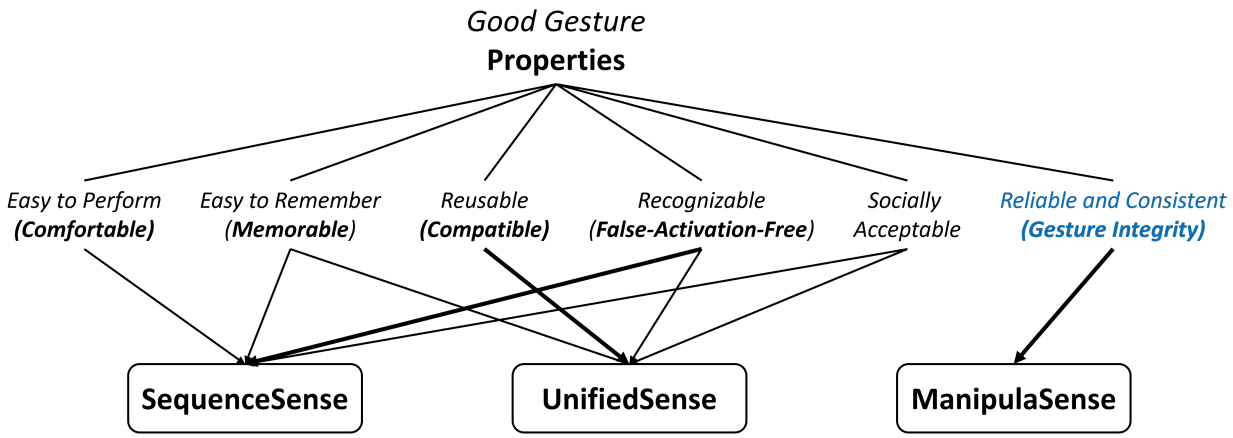


Figure 7.1: Revisiting the properties of a *Good Gesture* by illustrating how each of the properties is connected to this thesis.

Figure 7.1.

Finally, I introduced gesture integrity as a new property in the gesture design process to facilitate users' reliable and consistent gestural interactions. To ensure gesture integrity during continuous hand movements in VR, I proposed ManipulaSense, which will eventually help users get the status of their performed gestures. This new property inclusion is also shown in Figure 7.1. This new property is not only applicable to VR situations but also to situations where intruders inject IMU signals to change the outcome of the performed gestures [327, 328]. However, ensuring the integrity of performed gestures to avoid IMU signal injections is out of the scope of this research.

Although these three components operate independently, they can be interconnected to enhance usable gesture interactions. For instance, SequenceSense could detect gestures by sequencing atomic actions, which could then be integrated with UnifiedSense to make gestures more adaptable for general users. Since gestures designed by designers may not always suit general users' preferences, combining these modules could allow users to customize gestures, thereby increasing gesture adaptability. Lastly, I can incorporate a model akin to ManipulaSense to safeguard gesture integrity during real-world usage and examine potential IMU signal injections, which may warrant further investigation.

Here, I summarize the solutions and the findings from the experiments. This chapter concludes by providing directions for future research.

7.1 Summary

7.1.1 Designing Conflict-Free Body-Based Gestures

- In Chapter 3, I presented *SequenceSense*, a framework developed to help designers efficiently build a robust body-based gesture set using a sequence-based gesture recognizer and an automatic conflict analyzer. Instead of training a gesture recognizer using raw features, SequenceSense detects atomic actions in a gesture and uses their sequence to define and recognize gestures. This allows the gesture designer to easily modify the gestures by slightly modifying atomic actions in the gesture sequence and to immediately check the recognition performance and potential false activations without re-collecting the gesture samples or conducting experiments. I believe that this will allow gesture designers to focus more on the usability of gestures and be less concerned about their robustness.

- I evaluated the feasibility of our system with foot gestures, which are shown to be preferred over other body-based gestures in scenarios where users cannot use their hands [9]. I adopted 17 foot gestures from prior work on body-based and foot gestures [25, 9] that focused on usability.

- In our evaluation with 12 participants, SequenceSense showed higher gesture classification accuracy (95%) to that of the machine-learning-based gesture recognizer that uses statistical features (90%), while keeping false positive rates during daily activities to be approximately 58.5%, which is lower than that of ML-based recognizer (70%). The false-positive rate of 58.5% would cause approximately 59 false activations per hour in our dataset collected from two days of daily activities, which is still high. I used the gesture conflict analysis provided by SequenceSense to identify gestures that are similar to daily activities and slightly modified them to avoid conflicts. The modified gestures could reduce the average number of false activations per hour to 2.3, which is more than 96% lower than the original gestures.

- I conducted a user study to validate our tool in effectively designing gestures and compare its effectiveness over MAGIC [1]. I measured the quality of the designed gestures, the efficiency of designing gestures, and the overall user experience. The results from our user study showed that using SequenceSense, participants were able to design usable gestures with shorter gesture sequences, requiring approximately half of the time compared to MAGIC. Moreover, all the participants in our study preferred using SequenceSense over MAGIC to design gestures as gesture design with SequenceSense requires less effort and especially does not require participants to re-collect modified gesture samples when conflicts occur.

- Finally, SequenceSense is a new gesture design tool that uses the sequence of atomic actions and an instant gesture performance analysis to help gesture designers efficiently design, test, and modify usable foot-based gestures by minimizing the number of user studies.

7.1.2 Making Gestures Compatibility Across Devices

- To make gestures reusable and compatible across devices, I presented *UnifiedSense* in Chapter 4. UnifiedSense is a system that utilizes the input gesture recognized on a wearable device (primary device) and the sensor data from all other connected wearable devices (secondary devices) to train a unified gesture recognition model to enable interactions for the primary device even when the device is absent. Using the *over-the-shoulder training* method, the UnifiedSense system trains the model by itself by collecting training samples as users perform gestures on wearable devices. I assume that a wearable device can reliably recognize input gestures designed for it (e.g., touch gestures on a smartwatch), and the recognition result can be used for labeling a gesture training sample. As the user continues using gestures on their wearable devices, the unified gesture recognizer will be trained automatically over time.

- I verified the UnifiedSense system by conducting a user study with 15 participants. UnifiedSense collected sensor data from four wearable devices (headphones, smartwatch, smart ring, and smart glasses), and the participants performed five touch gestures (tap and four directional swipes) on the devices. The experiment results showed that, if a user performs gestures for the first time, the system can only detect gestures at an average accuracy of 81.01% using a model trained by other users wearing the same configuration of devices. However, if the new user continues to use the gestures, after the gesture set was used 20 times by themselves, the recognition accuracy increases to 95.2%.

- To address the usage of the UnifiedSense system *in the wild* by alleviating the challenge of eliminating false gesture activations associated with natural body movements, I implemented a binary classifier to differentiate intended gestures and natural body movements. I collected 20 hours of regular activity data from 12 users. When tested with artificially generated real-time data that include both natural movements and intentional gestures, UnifiedSense could recognize the gestures at an average accuracy of 90.9% with a gesture recognizer trained with 22 samples per gesture.

- I developed an *Android app* that shows the readiness of the model (i.e., the likelihood of properly recognizing gestures) so that the user can know if they can start using the gestures without the primary device.

- Finally, UnifiedSense is a novel mechanism to train the gesture recognition model using redundant sensor data collected from secondary devices while users perform gestures on primary devices using the over-the-shoulder training method.

7.1.3 Investigation of Hand-Movements Manipulation

- In this thesis, I was more interested in building a method that would ensure gesture integrity by providing feedback to the users on their natural hand movements. However, before deep-diving into building the solution, I first investigated the impacts of manipulations by carefully examining the kinematic metrics of users' hand movement. Specifically, I explored the impact of perception manipulation in VR on the relationship between motor learning and proprioception, particularly in adapting to kinematic changes, with a focus on the implications for hand movement and proprioception [230].

- I investigated the effect of two commonly used perception manipulation techniques for hand interaction, orientation, and magnitude manipulation techniques, on the performance and movement behavior during reaching tasks in Chapter 5. I conducted two user studies to separately investigate the effect of orientation and magnitude manipulation techniques in VR.

- For both cases, our results revealed that the task completion time remains consistent in the presence of manipulations. However, I observed that the movement behavior changed after participants were exposed to perception manipulations, and the behavior difference was more prominent when the level of manipulation was higher.

- The most significant difference in the movement behavior was the quality of proprioceptive movement, also known as ballistic movement. The distance to the target after the ballistic movement was significantly higher with the presence of perception manipulation, indicating the participants had to make more corrective movements to properly reach the target.

- While participants used faster corrective movement to make up for the error in the ballistic movement, which resulted in a similar overall task completion time, it may require additional effort and coordination from the users.

- This increased effort may lead to higher cognitive load, as individuals need to adjust their motor control strategy in real time to compensate for the manipulations. Higher cognitive load can potentially interfere with the execution of the movement and may negatively impact overall motor performance [230].

- Finally, I performed an in-depth investigation of the effect of orientation and magnitude manipulation techniques on the user's hand movement behavior in reaching tasks in VR through two user studies.

7.1.4 Detecting Hand-Movements Manipulation

- Since hand manipulation risks degrading task performance and, if maliciously applied, poses a threat to user safety, I advocate for a user's prerogative to be informed of any such potential manipulations before application usage. To enhance the integrity of hand movements in VR, I introduced *ManipulaSense* in Chapter 3. ManipulaSense is an *Autoencoder*-based anomaly detection technique that leverages users' inherent hand movements to identify perception manipulation.

- I trained ManipulaSense on regular (i.e., non-manipulated) hand movement patterns and employed a stochastic thresholding approach for anomaly detection.

- I validated ManipulaSense through a technical evaluation involving 21 participants engaged in reaching tasks under manipulated and non-manipulated scenarios.

- The results demonstrated a high accuracy of perception manipulation detection at 93.7%, with an F1-score of 93.9%.

- To our knowledge, this is the first study to specifically address the detection of perception manipulations in VR, establishing a foundational contribution to the field.

- Finally, I made the dataset used to build ManipulaSense public to facilitate future research, addressing the absence of publicly available datasets for perception manipulation research.

7.2 Directions for Future Research

Research is a continuous endeavor, offering ample scope for further improvements and experimentation. The solutions proposed in this dissertation are efficient and effective, yet they possess limitations that carve avenues for future research. Below, I outline potential directions for future research endeavors.

- **Cross-Modal Sensor Fusion for Enhanced Gesture Recognition:** I envision developing a comprehensive sensor fusion model by leveraging the insights from UnifiedSense's use of IMUs and exploring the integration of additional sensors across various wearables. This model would aim to enhance gesture recognition across a broader spectrum of body movements, addressing SequenceSense's challenges in segmenting gestures into atomic actions without relying on specific behaviors like foot-ground contact. Incorporating diverse sensor inputs could facilitate more accurate segmentation and recognition of subtle gestures beyond foot movements.

- **Unified Gesture Vocabulary Across Devices and Environments:** I envision developing a unified gesture vocabulary that can be recognized consistently across different devices and settings, including VR. This involves creating a framework that not only recognizes gestures made with different parts of the body but also understands the context of these gestures within various environments, potentially enhancing user experience in both physical and virtual settings.

- **Gesture Usability and Adaptation:** Building on the usability prediction challenges identified in SequenceSense, future research should focus on developing adaptive models that can predict the ease of use and potential fatigue associated with compound gestures. This involves creating a dynamic model that considers the variability in gesture execution (e.g., changes in user behavior over time or in different contexts, as highlighted by ManipulaSense). Such a model would be invaluable in tailoring gesture-based interfaces to individual user needs, enhancing long-term usability and comfort.

- **Composite and Contextual Gesture Recognition:** Addressing the composite gesture recognition challenges from UnifiedSense and the specific manipulation detection from ManipulaSense, future work should investigate advanced recognition techniques that can accurately identify complex gestures and manipulations. This includes exploring hierarchical and sequence-based models that can understand the context and sequence of gestures, potentially enabling more sophisticated interaction paradigms in both physical and virtual environments.

- **Cross-Device and Cross-Environment Gesture Interaction:** Finally, I envision a cross-device, cross-environment gesture interaction ecosystem that seamlessly integrates the strengths of SequenceSense, UnifiedSense, and ManipulaSense. This ecosystem would allow for a fluid interaction experience, where gestures initiated on one device could be continued or complemented by gestures on another across both physical and virtual spaces. Investigating the design of such an ecosystem requires a deep understanding of the interplay between gesture recognition technologies, user interface design, and user experience optimization.

- **Enhancing Gesture Recognition Accuracy through UnifiedSense:** A promising avenue for future research lies in the further development and application of the UnifiedSense framework to address and correct gesture mis-detections. This endeavor involves harnessing sensor data from secondary devices to improve the accuracy of primary gesture recognition systems, which often exhibit limitations in reliably detecting user interactions. The observed discrepancies, such as the mis-detection of a wipe-right gesture as a tap, underscore the potential for UnifiedSense to significantly enhance user experience by ensuring

that gestures are interpreted correctly, even in the presence of sensing inaccuracies by the primary device.

Future studies should focus on refining the mechanisms through which UnifiedSense leverages secondary sensor inputs to compensate for mis-detections. This includes exploring adaptive algorithms that can dynamically interpret sensor data from a range of devices to provide a more accurate and reliable recognition of user gestures. Moreover, investigating the integration of machine learning models capable of understanding the context and nuances of different gestures could further improve the robustness of gesture recognition systems. The potential impact of this research extends beyond merely correcting mis-detections; it promises to enhance the way we interact with technology by creating a more intuitive and error-resilient interface. By enhancing the fidelity of gesture recognition, we can significantly reduce user frustration and increase satisfaction, paving the way for more seamless and natural human-computer interactions.

7.3 Appendix

7.3.1 Accepted Publications

1. UnifiedSense: Enabling Without-Device Gesture Interactions Using Over-the-shoulder Training Between Redundant Wearable Sensors

   Md Aashikur Rahman Azim, Adil Rahman, and Seongkook Heo

   PACM, MobileHCI, Athens, Greece, 2023

2. SequenceSense: A Tool for Designing Usable Foot-Based Gestures Using a Sequence-Based Gesture Recognizer

   Md Aashikur Rahman Azim, Adil Rahman, and Seongkook Heo

   International Journal of Human-Computer Studies, Elsevier, 2023

3. Take My Hand: Automated Hand-Based Spatial Guidance for the Visually Impaired

   Adil Rahman, **Md Aashikur Rahman Azim**, and Seongkook Heo

   In Proc. of CHI, ACM, Hamburg, Germany, 2023

   Best Paper Award

4. Over-The-Shoulder Training Between Redundant Wearable Sensors for Unified Gesture Interactions

   Md Aashikur Rahman Azim, Adil Rahman, and Seongkook Heo

   In Proc. of UIST Adjunct, Oregon, 2022

5. FluidMeet: Enabling Frictionless Transitions Between In-Group, Between-Group, and Private Conversations During Virtual Breakout Meetings.

   Erzhen Hu, **Md Aashikur Rahman Azim**, and Seongkook Heo

   In Proc. of CHI, ACM, Louisiana, 2022

7.3.2 Under Review

1. Your Hands Can Tell: Detecting Perception Manipulation in VR Using Hand Movement

   Md Aashikur Rahman Azim, Zihao Su, and Seongkook Heo

   ISS, 2024

2. "It Feels Like I am Invited to Communicate": Mediating Ad-Hoc Bystander-VR User Interruptions Through Proactive Proxies

   Adil Rahman*, Wen Ying*, **Md Aashikur Rahman Azim***, Michelle Annett, and Seongkook Heo

   DIS, 2024 (*equal contribution)

3. ThingMoji: Object-based Emojis Captured from Live Stream to Support In-Stream Visual Communication

   Erzhen Hu, Qian Wan, Changkong Zhou, **Md Aashikur Rahman Azim**, PiaoHong Wang, Xingyi Hu, Yuhan Zeng, Zhicong Lu, and Seongkook Heo

   CSCW, 2024

4. Investigating the Impacts of Perception Manipulation in VR on Proprioceptive Movement during Reaching Task

   Md Aashikur Rahman Azim, Zihao Su, Peiyu Zhang, and Seongkook Heo

   (Under Preparation)

Bibliography

[1] Daniel Ashbrook and Thad Starner. Magic: a motion gesture design tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2159–2168, 2010.

[2] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2114–2124, 2021.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[4] Daniel L Ashbrook. *Enabling mobile microinteractions*. Georgia Institute of Technology, 2010.

[5] Christian Loclair, Sean Gustafson, and Patrick Baudisch. Pinchwatch: a wearable device for one-handed microinteractions. In *Proc. MobileHCI*, volume 10. Citeseer, 2010.

[6] Vivian Genaro Motti and Kelly Caine. Micro interactions and multi dimensional graphical user interfaces in the design of wrist worn wearables. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 59, pages 1712–1716. SAGE Publications Sage CA: Los Angeles, CA, 2015.

[7] Jonggi Hong, Seongkook Heo, Poika Isokoski, and Geehyuk Lee. Splitboard: A simple split soft keyboard for wristwatch-sized touch screens. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 1233–1236, 2015.

[8] Gierad Laput, Robert Xiao, and Chris Harrison. ViBand: High-Fidelity Bio-Acoustic Sensing Using Commodity Smartwatch Accelerometers. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 321–333, Tokyo Japan, October 2016. ACM.

[9] Seongkook Heo, Michelle Annett, Benjamin J Lafreniere, Tovi Grossman, and George W Fitzmaurice. No need to stop what you're doing: Exploring no-handed smartwatch interaction. In *Graphics Interface*, pages 107–114, 2017.

[10] Jeremy Scott, David Dearman, Koji Yatani, and Khai N Truong. Sensing foot gestures from the pocket. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 199–208. ACM, 2010.

[11] Andrew Crossan, Mark McGill, Stephen Brewster, and Roderick Murray-Smith. Head tilting for interaction in mobile contexts. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 1–10, 2009.

[12] Andrew Crossan, Stephen Brewster, and Alexander Ng. Foot tapping for mobile interaction. *Proceedings of HCI 2010 24*, pages 418–422, 2010.

[13] T Scott Saponas, Desney S Tan, Dan Morris, Ravin Balakrishnan, Jim Turner, and James A Landay. Enabling always-available input with muscle-computer interfaces. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 167–176, 2009.

[14] Chris Harrison, Desney Tan, and Dan Morris. Skinput: appropriating the body as an input surface. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 453–462, 2010.

[15] Brian Amento, Will Hill, and Loren Terveen. The sound of one hand: A wrist-mounted bio-acoustic fingertip gesture interface. In *CHI'02 Extended Abstracts on Human Factors in Computing Systems*, pages 724–725, 2002.

[16] Travis Deyle, Szabolcs Palinko, Erika Shehan Poole, and Thad Starner. Hambone: A bio-acoustic gesture interface. In *2007 11th IEEE International Symposium on Wearable Computers*, pages 3–10. IEEE, 2007.

[17] Simon T Perrault, Eric Lecolinet, James Eagan, and Yves Guiard. Watchit: simple gestures and eyes-free interaction for wristwatches and bracelets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1451–1460, 2013.

[18] Da-Yuan Huang, Liwei Chan, Shuo Yang, Fan Wang, Rong-Hao Liang, De-Nian Yang, Yi-Ping Hung, and Bing-Yu Chen. Digitspace: Designing thumb-to-fingers touch interfaces for one-handed and eyes-free interactions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 1526–1537, 2016.

[19] Cheng Zhang, Xiaoxuan Wang, Anandghan Waghmare, Sumeet Jain, Thomas Ploetz, Omer T Inan, Thad E Starner, and Gregory D Abowd. Fingorbits: interaction with wearables using synchronized thumb movements. In *Proceedings of the 2017 ACM International Symposium on Wearable Computers*, pages 62–65, 2017.

[20] William Antonelli and Jennifer Still. How to use Spotify on your Apple Watch to play music or control playback. https://www.businessinsider.in/tech/how-to/how-to-use-spotify-on-your-apple-watch-to-play-music-or-control-playback/articleshow/82538864.cms, 2021. Online; accessed May 2023.

[21] Sunggeun Ahn, Seongkook Heo, and Geehyuk Lee. Typing on a smartwatch for smart glasses. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*, pages 201–209, 2017.

[22] Pei-Yu Chi and Yang Li. Weave: Scripting cross-device wearable interaction. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pages 3923–3932, 2015.

[23] Daniel Wigdor and Dennis Wixon. *Brave NUI world: designing natural user interfaces for touch and gesture*. Elsevier, 2011.

[24] Miguel A Nacenta, Yemliha Kamber, Yizhou Qiang, and Per Ola Kristensson. Memorability of pre-designed and user-defined gesture sets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing systems*, pages 1099–1108, 2013.

[25] Yasmin Felberbaum and Joel Lanir. Better understanding of foot gestures: An elicitation study. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 334. ACM, 2018.

[26] Jacob O Wobbrock, Meredith Ringel Morris, and Andrew D Wilson. User-defined gestures for surface computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1083–1092. ACM, 2009.

[27] Hayati Havlucu, Mehmet Yarkın Ergin, Idil Bostan, Oğuz Turan Buruk, Tilbe Göksun, and Oğuzhan Özcan. It made more sense: Comparison of user-elicited on-skin touch and freehand gesture sets. In *Distributed, Ambient and Pervasive Interactions: 5th International Conference, DAPI 2017, Held as Part of HCI International 2017, Vancouver, BC, Canada, July 9–14, 2017, Proceedings 5*, pages 159–171. Springer, 2017.

[28] Huixiang Zhang, Wenteng Xu, Chunlei Chen, Liang Bai, and Yonghui Zhang. Your knock is my command: Binary hand gesture recognition on smartphone with accelerometer. *Mobile Information Systems*, 2020, 2020.

[29] Tomoya Murata and Jungpil Shin. Hand gesture and character recognition based on kinect sensor. *International Journal of Distributed Sensor Networks*, 10(7):278460, 2014.

[30] Koumei Fukahori, Daisuke Sakamoto, and Takeo Igarashi. Exploring subtle foot plantar-based gestures with sock-placed pressure sensors. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3019–3028. ACM, 2015.

[31] Hao Lu and Yang Li. Gesture on: Enabling always-on touch gestures for fast mobile access from the device standby mode. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3355–3364, 2015.

[32] Don Norman. *The design of everyday things: Revised and expanded edition*. Basic books, 2013.

[33] Emily Dao, Andreea Muresan, Kasper Hornbæk, and Jarrod Knibbe. Bad breakdowns, useful seams, and face slapping: Analysis of vr fails on youtube. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2021.

[34] Peter Casey, Ibrahim Baggili, and Ananya Yarramreddy. Immersive virtual reality attacks and the human joystick. *IEEE Transactions on Dependable and Secure Computing*, 18(2):550–562, 2019.

[35] Wen-Jie Tseng, Elise Bonnail, Mark Mcgill, Mohamed Khamis, Eric Lecolinet, Samuel Huron, and Jan Gugenheimer. The dark side of perceptual manipulations in virtual reality. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2022.

[36] Aman Parnami, Apurva Gupta, Gabriel Reyes, Ramik Sadana, Yang Li, and Gregory D Abowd. Mogeste: A mobile tool for in-situ motion gesture design. In *Proceedings of the 8th Indian Conference on Human Computer Interaction*, pages 35–43, 2016.

[37] Xuhai Xu, Haitian Shi, Xin Yi, Wenjia Liu, Yukang Yan, Yuanchun Shi, Alex Mariakakis, Jennifer Mankoff, and Anind K Dey. Earbuddy: Enabling on-face interaction via wireless earbuds. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2020.

[38] T Scott Saponas, Desney S Tan, Dan Morris, and Ravin Balakrishnan. Demonstrating the feasibility of using forearm electromyography for muscle-computer interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 515–524, 2008.

[39] Gilles Bailly, Jörg Müller, Michael Rohs, Daniel Wigdor, and Sven Kratz. Shoesense: a new perspective on gestural interaction and wearable applications. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1239–1248, 2012.

[40] Andrew Crossan, John Williamson, Stephen Brewster, and Rod Murray-Smith. Wrist rotation for interaction in mobile contexts. In *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, pages 435–438, 2008.

[41] Enrico Costanza, Samuel A Inverso, and Rebecca Allen. Toward subtle intimate interfaces for mobile devices using an emg controller. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 481–489, 2005.

[42] Mahfuz Rahman, Sean Gustafson, Pourang Irani, and Sriram Subramanian. Tilt techniques: investigating the dexterity of wrist-based input. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1943–1952, 2009.

[43] Jun Gong, Xing-Dong Yang, and Pourang Irani. Wristwhirl: One-handed continuous smartwatch input using wrist gestures. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, page 861–872, New York, NY, USA, 2016. Association for Computing Machinery.

[44] Paul Strohmeier, Roel Vertegaal, and Audrey Girouard. With a flick of the wrist: stretch sensors as lightweight input for mobile devices. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*, pages 307–308, 2012.

[45] Theophanis Tsandilas, Emmanuel Dubois, and Mathieu Raynal. Modeless pointing with low-precision wrist movements. In *Human-Computer Interaction–INTERACT 2013: 14th IFIP TC 13 International Conference, Cape Town, South Africa, September 2-6, 2013, Proceedings, Part III 14*, pages 494–511. Springer, 2013.

[46] Faizan Haque, Mathieu Nancel, and Daniel Vogel. Myopoint: Pointing and clicking using forearm mounted electromyography and inertial motion sensors. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 3653–3656, 2015.

[47] Eleanor Jones, Jason Alexander, Andreas Andreou, Pourang Irani, and Sriram Subramanian. Gestext: accelerometer-based gestural text-entry systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2173–2182, 2010.

[48] Kurt Partridge, Saurav Chatterjee, Vibha Sazawal, Gaetano Borriello, and Roy Want. Tilttype: accelerometer-supported text entry for very small devices. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 201–204, 2002.

[49] Daniel Wigdor and Ravin Balakrishnan. Tilttext: using tilt for text input to mobile phones. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 81–90, 2003.

[50] Hui-Shyong Yeo, Xiao-Shen Phang, Steven J Castellucci, Per Ola Kristensson, and Aaron Quigley. Investigating tilt-based gesture keyboard entry for single-handed text entry on large devices. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 4194–4202, 2017.

[51] Aakar Gupta, Cheng Ji, Hui-Shyong Yeo, Aaron Quigley, and Daniel Vogel. RotoSwype: Word-Gesture Typing using a Ring. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, Glasgow Scotland Uk, May 2019. ACM.

[52] Xiang 'Anthony' Chen, Tovi Grossman, Daniel J Wigdor, and George Fitzmaurice. Duet: exploring joint interactions on a smart phone and a smart watch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 159–168, 2014.

[53] Andrea Bianchi and Seungwoo Je. Disambiguating touch with a smart-ring. In *Proceedings of the 8th Augmented Human International Conference*, pages 1–5, Silicon Valley California USA, March 2017. ACM.

[54] Gerard Wilkinson, Ahmed Kharrufa, Jonathan Hook, Bradley Pursglove, Gavin Wood, Hendrik Haeuser, Nils Y Hammerla, Steve Hodges, and Patrick Olivier. Expressy: Using a wrist-worn inertial measurement unit to add expressiveness to touch-based interactions. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, pages 2832–2844, 2016.

[55] Roy Shilkrot, Jochen Huber, Jürgen Steimle, Suranga Nanayakkara, and Pattie Maes. Digital digits: A comprehensive survey of finger augmentation devices. *ACM Computing Surveys (CSUR)*, 48(2):1–29, 2015.

[56] Ashley Colley, Virve Inget, Inka Rantala, and Jonna Häkkilä. Investigating interaction with a ring form factor. In *Proceedings of the 16th International Conference on Mobile and Ubiquitous Multimedia*, pages 107–111, 2017.

[57] Andrew M Webb, Michel Pahud, Ken Hinckley, and Bill Buxton. Wearables as context for guiard-abiding bimanual touch. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 287–300, 2016.

[58] Xing-Dong Yang, Tovi Grossman, Daniel Wigdor, and George Fitzmaurice. Magic finger: always-available input through finger instrumentation. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 147–156, 2012.

[59] Lee Stearns, Uran Oh, Leah Findlater, and Jon E Froehlich. Touchcam: Realtime recognition of location-specific on-body gestures to support users with visual impairments. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):1–23, 2018.

[60] Martin Weigel and Jürgen Steimle. Deformwear: Deformation input on tiny wearable devices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(2):1–23, 2017.

[61] Huy Viet Le, Valentin Schwind, Philipp Göttlich, and Niels Henze. Predictouch: A system to reduce touchscreen latency using neural networks and inertial measurement units. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*, pages 230–239, 2017.

[62] Alessandra Moschetti, Laura Fiorini, Dario Esposito, Paolo Dario, and Filippo Cavallo. Recognition of daily gestures with wearable inertial rings and bracelets. *Sensors*, 16(8):1341, 2016.

[63] Gabriel Reyes, Jason Wu, Nikita Juneja, Maxim Goldshtein, W Keith Edwards, Gregory D Abowd, and Thad Starner. Synchrowatch: One-handed synchronous smartwatch gestures using correlation and magnetic sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):1–26, 2018.

[64] Stephen Brewster, Joanna Lumsden, Marek Bell, Malcolm Hall, and Stuart Tasker. Multimodal'eyes-free'interaction techniques for wearable devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 473–480, 2003.

[65] Augusto Esteves, Eduardo Velloso, Andreas Bulling, and Hans Gellersen. Orbits: Gaze interaction for smart watches using smooth pursuit eye movements. In *Proceedings of the 28th annual ACM symposium on user interface software & technology*, pages 457–466, 2015.

[66] Dmitry O Gorodnichy and Gerhard Roth. Nouse 'use your nose as a mouse'perceptual vision technology for hands-free games and interfaces. *Image and Vision Computing*, 22(12):931–942, 2004.

[67] Ondrej Polacek, Thomas Grill, and Manfred Tscheligi. Nosetapping: what else can you do with your nose? In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, pages 1–9, 2013.

[68] Adam Zarek, Daniel Wigdor, and Karan Singh. Snout: One-handed use of capacitive touch devices. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 140–147, 2012.

[69] Chris Salem and Shumin Zhai. An isometric tongue pointing device. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, pages 538–539, 1997.

[70] T Scott Saponas, Daniel Kelly, Babak A Parviz, and Desney S Tan. Optically sensing tongue gestures for computer input. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 177–180, 2009.

[71] Shwetak N Patel and Gregory D Abowd. Blui: low-cost localized blowable user interfaces. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 217–220, 2007.

[72] Wei-Hung Chen. Blowatch: Blowable and hands-free interaction for smartwatches. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on human factors in computing systems*, pages 103–108, 2015.

[73] Oleg Špakov, Poika Isokoski, and Päivi Majaranta. Look and lean: accurate head-assisted eye pointing. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 35–42, 2014.

[74] Oleg Špakov and Päivi Majaranta. Enhanced gaze interaction using simple head gestures. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 705–710, 2012.

[75] Javier Varona, Cristina Manresa-Yee, and Francisco J Perales. Hands-free vision-based interface for computer accessibility. *Journal of Network and Computer Applications*, 31(4):357–374, 2008.

[76] Louis-Philippe Morency and Trevor Darrell. Head gesture recognition in intelligent interfaces: the role of context in improving recognition. In *Proceedings of the 11th international conference on Intelligent user interfaces*, pages 32–38, 2006.

[77] Tomi Nukarinen, Jari Kangas, Oleg Špakov, Poika Isokoski, Deepak Akkil, Jussi Rantala, and Roope Raisamo. Evaluation of headturn: An interaction technique using the gaze and head turns. In *Proceedings of the 9th Nordic Conference on Human-Computer Interaction*, pages 1–8, 2016.

[78] Zhenyu Tang, Chenyu Yan, Sijie Ren, and Huagen Wan. Headpager: page turning with computer vision based head interaction. In *Computer Vision–ACCV 2016 Workshops: ACCV 2016 International Workshops, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part III 13*, pages 249–257. Springer, 2017.

[79] Shanhe Yi, Zhengrui Qin, Ed Novak, Yafeng Yin, and Qun Li. Glassgesture: Exploring head gesture interface of smart glasses. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[80] Augusto Esteves, David Verweij, Liza Suraiya, Rasel Islam, Youryang Lee, and Ian Oakley. Smoothmoves: Smooth pursuits head movements for augmented reality. In *Proceedings of the 30th annual acm symposium on user interface software and technology*, pages 167–178, 2017.

[81] Mikko Kytö, Barrett Ens, Thammathip Piumsomboon, Gun A Lee, and Mark Billinghurst. Pinpointing: Precise head-and eye-based target selection for augmented reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2018.

[82] Shahram Jalaliniya, Diako Mardanbeigi, Thomas Pederson, and Dan Witzner Hansen. Head and eye movement as pointing modalities for eyewear computers. In *2014 11th International Conference on Wearable and Implantable Body Sensor Networks Workshops*, pages 50–53. IEEE, 2014.

[83] Shahram Jalaliniya, Diako Mardanbegi, and Thomas Pederson. Magic pointing for eyewear computers. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers*, pages 155–158, 2015.

[84] Richard A Monty and John W Senders. *Eye movements and psychological processes*. Routledge, 2017.

[85] Rob Jacob and Sophie Stellmach. What you look at is what you get: gaze-based user interfaces. *interactions*, 23(5):62–65, 2016.

[86] B Biguer, Marc Jeannerod, and Claude Prablanc. The coordination of eye, head, and arm movements during reaching at a single visual target. *Experimental brain research*, 46(2):301–304, 1982.

[87] Diako Mardanbegi, Dan Witzner Hansen, and Thomas Pederson. Eye-based head gestures. In *Proceedings of the symposium on eye tracking research and applications*, pages 139–146, 2012.

[88] Glenn Pearson and Mark Weiser. Exploratory evaluation of a planar foot-operated cursor-positioning device. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 13–18. ACM, 1988.

[89] Yuichiro Takeuchi. Gilded gait: reshaping the urban experience with augmented footsteps. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 185–188. ACM, 2010.

[90] Denys JC Matthies, Franz Müller, Christoph Anthes, and Dieter Kranzlmüller. Shoesolesense: proof of concept for a wearable foot interface for virtual and real environments. In *Proceedings of the 19th ACM Symposium on Virtual Reality Software and Technology*, pages 93–96. ACM, 2013.

[91] Stefano Papetti, Federico Fontana, Marco Civolani, Amir Berrezag, and Vincent Hayward. Audio-tactile display of ground properties using interactive shoes. In *International Workshop on Haptic and Audio Interaction Design*, pages 117–128. Springer, 2010.

[92] KangKang Yin and Dinesh K Pai. Footsee: an interactive animation system. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 329–338. Eurographics Association, 2003.

[93] Zhihan Lv, Shengzhong Feng, Muhammad Sikandar Lal Khan, Shafiq Ur Réhman, and Haibo Li. Foot motion sensing: augmented game interface based on foot interaction for smartphone. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems*, pages 293–296. ACM, 2014.

[94] Volker Paelke, Christian Reimann, and Dirk Stichling. Kick-up menus. In *CHI'04 Extended Abstracts on Human Factors in Computing Systems*, pages 1552–1552. ACM, 2004.

[95] Teng Han, Jason Alexander, Abhijit Karnik, Pourang Irani, and Sriram Subramanian. Kick: investigating the use of kick gestures for mobile interactions. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, pages 29–32. ACM, 2011.

[96] Johannes Schöning, Florian Daiber, Antonio Krüger, and Michael Rohs. Using hands and feet to navigate and manipulate spatial data. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*, pages 4663–4668. ACM, 2009.

[97] Ricardo Jota, Pedro Lopes, Daniel Wigdor, and Joaquim Jorge. Let's kick it: how to stop wasting the bottom third of your large screen display. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 1411–1414. ACM, 2014.

[98] Daniel Horodniczy and Jeremy R Cooperstock. Free the hands! enhanced target selection via a variable-friction shoe. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 255–259. ACM, 2017.

[99] Insook Choi and Carlos Ricci. Foot-mounted gesture detection and its application in virtual environments. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, volume 5, pages 4248–4253. IEEE, 1997.

[100] Gerwin de Haan, Eric J Griffith, and Frits H Post. Using the wii balance board™ as a low-cost vr interaction device. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, pages 289–290. ACM, 2008.

[101] Florian Müller, Joshua McManus, Sebastian Günther, Martin Schmitz, Max Mühlhäuser, and Markus Funk. Mind the tap: Assessing foot-taps for interacting with head-mounted displays. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, page 477. ACM, 2019.

[102] Brian Williamson, C Wingrave, JJ LaViola, T Roberts, and P Garrity. Natural full body interaction for navigation in dismounted soldier training. In *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)*, 2011.

[103] Gabe Cohn, Daniel Morris, Shwetak Patel, and Desney Tan. Humantenna: using the body as an antenna for real-time whole-body interaction. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1901–1910, 2012.

[104] Xiang'Anthony' Chen. Body-centric interaction with mobile devices. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*, pages 385–386, 2012.

[105] Garth Shoemaker, Takayuki Tsukitani, Yoshifumi Kitamura, and Kellogg S Booth. Body-centric interaction techniques for very large wall displays. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, pages 463–472, 2010.

[106] Chris Harrison, Shilpa Ramamurthy, and Scott E Hudson. On-body interaction: armed and dangerous. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*, pages 69–76, 2012.

[107] Nadia Bianchi-Berthouze. Understanding the role of body movement in player engagement. *Human–Computer Interaction*, 28(1):40–75, 2013.

[108] Katherine Isbister, Rahul Rao, Ulf Schwekendiek, Elizabeth Hayward, and Jessamyn Lidasan. Is more movement better? a controlled comparison of movement-based games. In *Proceedings of the 6th international conference on Foundations of Digital Games*, pages 331–333, 2011.

[109] Jasmir Nijhar, Nadia Bianchi-Berthouze, and Gemma Boguslawski. Does movement recognition precision affect the player experience in exertion games? In *Intelligent Technologies for Interactive Entertainment: 4th International ICST Conference, INTETAIN 2011, Genova, Italy, May 25-27, 2011, Revised Selected Papers 4*, pages 73–82. Springer, 2012.

[110] Marco Pasch, Nadia Bianchi-Berthouze, Betsy van Dijk, and Anton Nijholt. Movement-based sports video games: Investigating motivation and gaming experience. *Entertainment computing*, 1(2):49–61, 2009.

[111] Jakob Tholander and Carolina Johansson. Design qualities for whole body interaction: learning from golf, skateboarding and bodybugging. In *Proceedings of the 6th nordic conference on human-computer interaction: Extending boundaries*, pages 493–502, 2010.

[112] Kathrin Gerling, Ian Livingston, Lennart Nacke, and Regan Mandryk. Full-body motion-based game interaction for older adults. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1873–1882, 2012.

[113] Katherine Isbister and Christopher DiMauro. Waggling the form baton: Analyzing body-movement-based design patterns in nintendo wii games, toward innovation of new possibilities for social and emotional experience. In *Whole body interaction*, pages 63–73. Springer, 2011.

[114] Dustin Freeman, Nathan LaPierre, Fanny Chevalier, and Derek Reilly. Tweetris: a study of whole-body interaction during a public art event. In *Proceedings of the 9th ACM Conference on Creativity & Cognition*, pages 224–233, 2013.

[115] Jörg Müller, Robert Walter, Gilles Bailly, Michael Nischt, and Florian Alt. Looking glass: a field study on noticing interactivity of a shop window. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 297–306, 2012.

[116] Radu-Daniel Vatavu. User-defined gestures for free-hand tv control. In *Proceedings of the 10th European conference on Interactive tv and video*, pages 45–48, 2012.

[117] Özge Samanci, Yanfeng Chen, and Ali Mazalek. Tangible comics: a performance space with full-body interaction. In *Proceedings of the international conference on advances in computer entertainment technology*, pages 171–178, 2007.

[118] Benjamin Schaeffer, Mark Flider, Hank Kaczmarski, Luc Vanier, Lance Chong, and Yu Hasegawa-Johnson. Tele-sports and tele-dance: full-body network interaction. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 108–116, 2003.

[119] Shlomo Berkovsky, Mac Coombe, and Richard Helmer. Activity interface for physical activity motivating games. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 273–276, 2010.

[120] Firaz Peer, Anne Friedlander, Ali Mazalek, and Florian'Floyd' Mueller. Evaluating technology that makes physical games for children more engaging. In *Proceedings of the 10th International Conference on Interaction Design and Children*, pages 193–196, 2011.

[121] Jeffrey Yim and TC Nicholas Graham. Using games to increase exercise motivation. In *Proceedings of the 2007 conference on Future Play*, pages 166–173, 2007.

[122] Johanna Höysniemi, Perttu Hämäläinen, and Laura Turkki. Wizard of oz prototyping of computer vision based action games for children. In *Proceedings of the 2004 conference on Interaction design and children: building a community*, pages 27–34, 2004.

[123] Andrew Macvean and Judy Robertson. Understanding exergame users' physical activity, motivation and behavior over time. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1251–1260, 2013.

[124] Hamilton A Hernandez, Zi Ye, TC Nicholas Graham, Darcy Fehlings, and Lauren Switzer. Designing action-based exergames for children with cerebral palsy. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1261–1270, 2013.

[125] Gazihan Alankus and Caitlin Kelleher. Reducing compensatory motions in video games for stroke rehabilitation. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 2049–2058, 2012.

[126] Christian Schönauer, Thomas Pintaric, and Hannes Kaufmann. Full body interaction for serious games in motor rehabilitation. In *Proceedings of the 2nd Augmented Human International Conference*, pages 1–8, 2011.

[127] Dean Harris Rubine. *The automatic recognition of gestures*. Carnegie Mellon University, 1991.

[128] R. B. Dannenberg and D. Amon. A gesture based user interface prototyping system. In *Proceedings of the 2nd Annual ACM SIGGRAPH Symposium on User Interface Software and Technology*, UIST '89, page 127–132, New York, NY, USA, 1989. Association for Computing Machinery.

[129] James A. Landay and Brad A. Myers. Extending an existing user interface toolkit to support gesture recognition. In *INTERACT '93 and CHI '93 Conference Companion on Human Factors in Computing Systems*, CHI '93, page 91–92, New York, NY, USA, 1993. Association for Computing Machinery.

[130] Allan Christian Long, James A. Landay, and Lawrence A. Rowe. Implications for a gesture design tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, page 40–47, New York, NY, USA, 1999. Association for Computing Machinery.

[131] Allen Cypher and Daniel Conrad Halbert. *Watch what I do: programming by demonstration*. MIT press, 1993.

[132] Allan Long, James Landay, and Lawrence Rowe. Quill?a gesture design tool for pen-based user interfaces. *U.C. Berkeley*, 01 2001.

[133] Björn Hartmann, Leith Abdulla, Manas Mittal, and Scott R Klemmer. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 145–154, 2007.

[134] Daniel Kohlsdorf, Thad Starner, and Daniel Ashbrook. Magic 2.0: A web tool for false positive prediction and prevention for gesture recognition systems. In *Face and Gesture 2011*, pages 1–6. IEEE, 2011.

[135] Ju-Whan Kim and Tek-Jin Nam. Eventhurdle: supporting designers' exploratory interaction prototyping with gesture-based sensors. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 267–276, 2013.

[136] Ju-Whan Kim, Han-Jong Kim, and Tek-Jin Nam. M. gesture: an acceleration-based gesture authoring system on multiple handheld and wearable devices. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2307–2318, 2016.

[137] Jeff K.T. Tang and Takeo Igarashi. CUBOD: A customized body gesture design tool for end users. In *Proceedings of the 27th International BCS Human Computer Interaction Conference (HCI 2013)*, 2013.

[138] Sujin Jang, Niklas Elmqvist, and Karthik Ramani. Gestureanalyzer: visual analytics for pattern analysis of mid-air hand gestures. In *Proceedings of the 2nd ACM symposium on Spatial user interaction*, pages 30–39, 2014.

[139] Philip Quinn, Seungyon Claire Lee, Melissa Barnhart, and Shumin Zhai. Active edge: Designing squeeze gestures for the google pixel 2. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2019.

[140] Jaime Ruiz and Yang Li. Doubleflip: a motion gesture delimiter for mobile interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2717–2720, 2011.

[141] Frederic Kerber, Philipp Schardt, and Markus Löchtefeld. Wristrotate: a personalized motion gesture delimiter for wrist-worn devices. In *Proceedings of the 14th international conference on mobile and ubiquitous multimedia*, pages 218–222, 2015.

[142] Bryan Wang and Tovi Grossman. Blyncsync: enabling multimodal smartwatch gestures with synchronous touch and blink. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2020.

[143] Adwait Sharma, Michael A. Hedderich, Divyanshu Bhardwaj, Bruno Fruchard, Jess McIntosh, Aditya Shekhar Nittala, Dietrich Klakow, Daniel Ashbrook, and Jürgen Steimle. Solofinger: Robust microgestures while grasping everyday objects. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2021.

[144] Ryo Kawahata, Atsushi Shimada, Takayoshi Yamashita, Hideaki Uchiyama, and Rin-ichiro Taniguchi. Design of a low-false-positive gesture for a wearable device. In *ICPRAM*, pages 581–588, 2016.

[145] Gierad Laput, Robert Xiao, and Chris Harrison. Viband: High-fidelity bio-acoustic sensing using commodity smartwatch accelerometers. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 321–333, 2016.

[146] Frederik Brudy, Christian Holz, Roman Rädle, Chi-Jui Wu, Steven Houben, Clemens Nylandsted Klokmose, and Nicolai Marquardt. Cross-device taxonomy: Survey, opportunities and challenges of interactions spanning across multiple devices. In *Proceedings of the 2019 chi conference on human factors in computing systems*, pages 1–28, 2019.

[147] Fitbit Inc. Fitbit Versa. https://www.fitbit.com/global/us/products/smartwatches/versa3, 2022. Online; accessed Aug 2022.

[148] Ōura Health Ltd. Oura Ring. https://ouraring.com/, 2022. Online; accessed Aug 2022.

[149] Google Inc. Google Glass. https://www.google.com/glass/start/, 2022. Online; accessed Aug 2022.

[150] Apple Inc. Apple – iOS 15 – Continuity. https://www.macrumors.com/guide/universal-control/, 2021. Online; accessed Aug 2022.

[151] Jun Gong, Yang Zhang, Xia Zhou, and Xing-Dong Yang. Pyro: Thumb-Tip Gesture Recognition Using Pyroelectric Infrared Sensing. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pages 553–563, Québec City QC Canada, October 2017. ACM.

[152] Jaime Lien, Nicholas Gillian, M. Emre Karagozler, Patrick Amihood, Carsten Schwesig, Erik Olson, Hakim Raja, and Ivan Poupyrev. Soli: ubiquitous gesture sensing with millimeter wave radar. *ACM Transactions on Graphics*, 35(4):1–19, July 2016.

[153] T. Scott Saponas, Desney S. Tan, Dan Morris, Ravin Balakrishnan, Jim Turner, and James A. Landay. Enabling always-available input with muscle-computer interfaces. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology - UIST '09*, page 167, Victoria, BC, Canada, 2009. ACM Press.

[154] Hongyi Wen, Julian Ramos Rojas, and Anind K. Dey. Serendipity: Finger Gesture Recognition using an Off-the-Shelf Smartwatch. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 3847–3851, San Jose California USA, May 2016. ACM.

[155] Artem Dementyev and Joseph A. Paradiso. WristFlex: low-power gesture input with wrist-worn pressure sensors. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 161–166, Honolulu Hawaii USA, October 2014. ACM.

[156] Rui Fukui, Masahiko Watanabe, Tomoaki Gyota, Masamichi Shimosaka, and Tomomasa Sato. Hand shape classification with a wrist contour sensor: development of a prototype device. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 311–314, 2011.

[157] Yasha Iravantchi, Yang Zhang, Evi Bernitsas, Mayank Goel, and Chris Harrison. Interferi: Gesture Sensing using On-Body Acoustic Interferometry. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13, Glasgow Scotland Uk, May 2019. ACM.

[158] Jess McIntosh, Asier Marzo, and Mike Fraser. SensIR: Detecting Hand Gestures with a Wearable Bracelet using Infrared Transmission and Reflection. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pages 593–597, Québec City QC Canada, October 2017. ACM.

[159] J. Rekimoto. GestureWrist and GesturePad: unobtrusive wearable interaction devices. In *Proceedings Fifth International Symposium on Wearable Computers*, pages 21–27, October 2001. ISSN: 1530-0811.

[160] Hoang Truong, Shuo Zhang, Ufuk Muncuk, Phuc Nguyen, Nam Bui, Anh Nguyen, Qin Lv, Kaushik Chowdhury, Thang Dinh, and Tam Vu. CapBand: Battery-free Successive Capacitance Sensing Wristband for Hand Gesture Recognition. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pages 54–67, Shenzhen China, November 2018. ACM.

[161] Yang Zhang and Chris Harrison. Tomo: Wearable, Low-Cost Electrical Impedance Tomography for Hand Gesture Recognition. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 167–173, Charlotte NC USA, November 2015. ACM.

[162] A.H.F. Lam, W.J. Li, Yunhui Liu, and Ning Xi. MIDS: micro input devices system using MEMS sensors. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1184–1189 vol.2, September 2002.

[163] Lei Jing, Zixue Cheng, Yinghui Zhou, Junbo Wang, and Tongjun Huang. Magic Ring: a self-contained gesture input device on finger. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia - MUM '13*, pages 1–4, Luleå, Sweden, 2013. ACM Press.

[164] Mehran Roshandel, Aarti Munjal, Peyman Moghadam, Shahin Tajik, and Hamed Ketabdar. Multisensor Finger Ring for Authentication Based on 3D Signatures. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Alfred Kobsa, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Demetri Terzopoulos, Doug Tygar, Gerhard Weikum, and Masaaki Kurosu, editors, *Human-Computer Interaction. Advanced Interaction Modalities and Techniques*, volume 8511, pages 131–138. Springer International Publishing, Cham, 2014. Series Title: Lecture Notes in Computer Science.

[165] Shahriar Nirjon, Jeremy Gummeson, Dan Gelb, and Kyu-Han Kim. TypingRing: A Wearable Ring Platform for Text Input. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 227–239, Florence Italy, May 2015. ACM.

[166] Bogdan-Florin Gheran, Radu-Daniel Vatavu, and Jean Vanderdonckt. Ring x2: Designing gestures for smart rings using temporal calculus. In *Proceedings of the 2018 ACM Conference Companion Publication on Designing Interactive Systems*, pages 117–122, 2018.

[167] Hsin-Ruey Tsai, Min-Chieh Hsiu, Jui-Chun Hsiao, Lee-Ting Huang, Mike Chen, and Yi-Ping Hung. TouchRing: subtle and always-available input using a multi-touch ring. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, pages 891–898, Florence Italy, September 2016. ACM.

[168] Jungmin Chung, Changhoon Oh, SoHyun Park, and Bongwon Suh. Pairring: a ring-shaped rotatable smartwatch controller. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–6, 2018.

[169] Guanhong Liu, Yizheng Gu, Yiwen Yin, Chun Yu, Yuntao Wang, Haipeng Mi, and Yuanchun Shi. Keep the Phone in Your Pocket: Enabling Smartphone Operation with an IMU Ring for Visually Impaired People. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(2):1–23, June 2020.

[170] Steven Feiner and Ari Shamash. Hybrid user interfaces: Breeding virtually bigger interfaces for physically smaller computers. In *Proceedings of the 4th annual ACM symposium on User interface software and technology*, pages 9–17, 1991.

[171] Shoya Ishimaru, Kai Kunze, Koichi Kise, Jens Weppner, Andreas Dengel, Paul Lukowicz, and Andreas Bulling. In the blink of an eye: combining head motion and eye blink frequency for activity recognition with google glass. In *Proceedings of the 5th augmented human international conference*, pages 1–4, 2014.

[172] Marcos Serrano, Barrett M Ens, and Pourang P Irani. Exploring the use of hand-to-face input for interacting with head-worn displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3181–3190, 2014.

[173] Ben Zhang, Yu-Hsiang Chen, Claire Tuna, Achal Dave, Yang Li, Edward Lee, and Björn Hartmann. Hobs: head orientation-based selection in physical spaces. In *Proceedings of the 2nd ACM symposium on Spatial user interaction*, pages 17–25, 2014.

[174] Shwetak N Patel, Jeffrey S Pierce, and Gregory D Abowd. A gesture-based authentication scheme for untrusted public terminals. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 157–160, 2004.

[175] David Dearman and Jeffery S Pierce. It's on my other computer! computing with multiple devices. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, pages 767–776, 2008.

[176] Stephanie Santosa and Daniel Wigdor. A field study of multi-device workflows in distributed workspaces. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 63–72, 2013.

[177] Simon Mayer and Gábor Sörös. User interface beaming–seamless interaction with smart things using personal wearable computers. In *2014 11th International Conference on Wearable and Implantable Body Sensor Networks Workshops*, pages 46–49. IEEE, 2014.

[178] Zihan Wang, Jiarong Li, Yuchao Jin, Jiyu Wang, Fang Yang, Gang Li, Xiaoyue Ni, and Wenbo Ding. Sensing beyond itself: Multi-functional use of ubiquitous signals towards wearable applications. *Digital Signal Processing*, 116:103091, 2021.

[179] Jess McIntosh, Asier Marzo, Mike Fraser, and Carol Phillips. Echoflex: Hand gesture recognition using ultrasound imaging. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 1923–1934, 2017.

[180] Gierad Laput and Chris Harrison. Sensing fine-grained hand activity with smartwatches. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2019.

[181] Manuel Meier, Paul Streli, Andreas Fender, and Christian Holz. Tapld: Rapid touch interaction in virtual reality using wearable sensing. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, pages 519–528. IEEE, 2021.

[182] Ken Hinckley. Synchronous gestures for multiple persons and computers. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 149–158, 2003.

[183] Ho-Man Colman Leung, Chi-Wing Fu, and Pheng-Ann Heng. Twistin: Tangible authentication of smart devices via motion co-analysis with a smartwatch. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(2):1–24, 2018.

[184] Jason Wu, Cooper Colglazier, Adhithya Ravishankar, Yuyan Duan, Yuanbo Wang, Thomas Ploetz, and Thad Starner. Seesaw: rapid one-handed synchronous gesture interface for smartwatches. In *Proceedings of the 2018 ACM International Symposium on Wearable Computers*, pages 17–20, 2018.

[185] Juyoung Lee, Shaurye Aggarwal, Jason Wu, Thad Starner, and Woontack Woo. Selfsync: exploring self-synchronous body-based hotword gestures for initiating interaction. In *Proceedings of the 23rd International Symposium on Wearable Computers*, pages 123–128, 2019.

[186] Google Inc. Android Wear: The developer's perspective. https://youtu.be/G7tXr-w35UA, 2014. Online; accessed Aug 2022.

[187] Mohamed Soliman, Franziska Mueller, Lena Hegemann, Joan Sol Roo, Christian Theobalt, and Jürgen Steimle. Fingerinput: Capturing expressive single-hand thumb-to-finger microgestures. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*, pages 177–187, 2018.

[188] Marcus Georgi, Christoph Amma, and Tanja Schultz. Recognizing hand and finger gestures with imu based motion and emg based muscle activity sensing. In *Biosignals*, pages 99–108. Citeseer, 2015.

[189] Jun Gong, Xing-Dong Yang, and Pourang Irani. WristWhirl: One-handed Continuous Smartwatch Input using Wrist Gestures. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 861–872, Tokyo Japan, October 2016. ACM.

[190] Wolf Kienzle and Ken Hinckley. Lightring: always-available 2d input on any surface. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 157–160, 2014.

[191] Jess McIntosh, Asier Marzo, and Mike Fraser. Sensir: Detecting hand gestures with a wearable bracelet using infrared transmission and reflection. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*, pages 593–597, 2017.

[192] Fang Hu, Peng He, Songlin Xu, Yin Li, and Cheng Zhang. Fingertrak: Continuous 3d hand pose tracking by deep learning hand silhouettes captured by miniature thermal cameras on wrist. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(2):1–24, 2020.

[193] David Kim, Otmar Hilliges, Shahram Izadi, Alex D Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. Digits: freehand 3d interactions anywhere using a wrist-worn gloveless sensor. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 167–176, 2012.

[194] Erwin Wu, Ye Yuan, Hui-Shyong Yeo, Aaron Quigley, Hideki Koike, and Kris M Kitani. Back-hand-pose: 3d hand pose estimation for a wrist-worn camera via dorsum deformation network. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 1147–1160, 2020.

[195] Xuhai Xu, Alexandru Dancu, Pattie Maes, and Suranga Nanayakkara. Hand range interface: Information always at hand with a body-centric mid-air input surface. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 1–12, 2018.

[196] Baptiste Caramiaux, Marco Donnarumma, and Atau Tanaka. Understanding gesture expressivity through muscle sensing. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 21(6):1–26, 2015.

[197] Yasha Iravantchi, Mayank Goel, and Chris Harrison. Beamband: Hand gesture sensing with ultrasonic beamforming. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–10, 2019.

[198] Ahmad Akl, Chen Feng, and Shahrokh Valaee. A Novel Accelerometer-Based Gesture Recognition System. *IEEE Transactions on Signal Processing*, 59(12):6197–6205, December 2011. Conference Name: IEEE Transactions on Signal Processing.

[199] Nicholas Gillian and Joseph A Paradiso. The gesture recognition toolkit. *The Journal of Machine Learning Research*, 15(1):3483–3487, 2014.

[200] Minwoo Kim, Jaechan Cho, Seongjoo Lee, and Yunho Jung. Imu sensor-based hand gesture recognition for human-machine interfaces. *Sensors*, 19(18):3827, 2019.

[201] Chao Xu, Parth H. Pathak, and Prasant Mohapatra. Finger-writing with Smartwatch: A Case for Finger and Hand Gesture Recognition using Smartwatch. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, pages 9–14, Santa Fe New Mexico USA, February 2015. ACM.

[202] Xuhai Xu, Jun Gong, Carolina Brum, Lilian Liang, Bongsoo Suh, Shivam Kumar Gupta, Yash Agarwal, Laurence Lindsey, Runchang Kang, Behrooz Shahsavari, et al. Enabling hand gesture customization on wrist-worn devices. In *CHI Conference on Human Factors in Computing Systems*, pages 1–19, 2022.

[203] Jiayang Liu, Zhen Wang, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. uwave: Accelerometer-based personalized gesture recognition. *TR0630-08, Rice University and Motorola Labs, Houston, Texas*, 2008.

[204] Thad Starner, Joshua Weaver, and Alex Pentland. A wearable computer based american sign language recognizer. In *Digest of Papers. First International Symposium on Wearable Computers*, pages 130–137. IEEE, 1997.

[205] Stephen J Mckenna and Kenny Morrison. A comparison of skin history and trajectory-based representation schemes for the recognition of user-specified gestures. *Pattern Recognition*, 37(5):999–1009, 2004.

[206] Hui-Shyong Yeo, Erwin Wu, Juyoung Lee, Aaron Quigley, and Hideki Koike. Opisthenar: Hand poses and finger tapping recognition by observing back of hand using embedded wrist camera. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, pages 963–971, 2019.

[207] Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020.

[208] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. *Advances in neural information processing systems*, 32, 2019.

[209] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[210] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[211] Konstantina Kilteni, Raphaela Groten, and Mel Slater. The sense of embodiment in virtual reality. *Presence: Teleoperators and Virtual Environments*, 21(4):373–387, 2012.

[212] Marta Matamala-Gomez, Tony Donegan, Sara Bottiroli, Giorgio Sandrini, Maria V Sanchez-Vives, and Cristina Tassorelli. Immersive virtual reality and virtual embodiment for pain relief. *Frontiers in human neuroscience*, 13:279, 2019.

[213] Michael I Posner, Mary J Nissen, and Raymond M Klein. Visual dominance: an information-processing account of its origins and significance. *Psychological review*, 83(2):157, 1976.

[214] Robert J Van Beers, Anne C Sittig, and Jan J Denier van der Gon. Integration of proprioceptive and visual position-information: An experimentally supported model. *Journal of neurophysiology*, 81(3):1355–1364, 1999.

[215] Robert J van Beers, Daniel M Wolpert, and Patrick Haggard. When feeling is more important than seeing in sensorimotor adaptation. *Current biology*, 12(10):834–837, 2002.

[216] David H Warren and Wallace T Cleaves. Visual-proprioceptive interaction under large amounts of conflict. *Journal of experimental psychology*, 90(2):206, 1971.

[217] Mahdi Azmandian, Mark Hancock, Hrvoje Benko, Eyal Ofek, and Andrew D Wilson. Haptic retargeting: Dynamic repurposing of passive haptics for enhanced virtual reality experiences. In *Proceedings of the 2016 chi conference on human factors in computing systems*, pages 1968–1979, 2016.

[218] Lung-Pan Cheng, Eyal Ofek, Christian Holz, Hrvoje Benko, and Andrew D Wilson. Sparse haptic proxy: Touch feedback in virtual environments using a general passive prop. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 3718–3728, 2017.

[219] Luv Kohli. Redirected touching: Warping space to remap passive haptics. In *2010 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 129–130. IEEE, 2010.

[220] Majed Samad, Elia Gatti, Anne Hermes, Hrvoje Benko, and Cesare Parise. Pseudo-haptic weight: Changing the perceived weight of virtual objects by manipulating control-display ratio. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2019.

[221] Sharif Razzaque. *Redirected walking*. The University of North Carolina at Chapel Hill, 2005.

[222] Frank Steinicke, Gerd Bruder, Jason Jerald, Harald Frenz, and Markus Lappe. Estimation of detection thresholds for redirected walking techniques. *IEEE transactions on visualization and computer graphics*, 16(1):17–27, 2009.

[223] Qi Sun, Anjul Patney, Li-Yi Wei, Omer Shapira, Jingwan Lu, Paul Asente, Suwen Zhu, Morgan McGuire, David Luebke, and Arie Kaufman. Towards virtual reality infinite walking: dynamic saccadic redirection. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018.

[224] Sharif Razzaque, David Swapp, Mel Slater, Mary C Whitton, and Anthony Steed. Redirected walking in place. In *EGVE*, volume 2, pages 123–130. Citeseer, 2002.

[225] Tiare Feuchtner and Jörg Müller. Extending the body for interaction with reality. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 5145–5157, 2017.

[226] Frank Steinicke, Gerd Bruder, Jason Jerald, Harald Frenz, and Markus Lappe. Analyses of human sensitivity to redirected walking. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, pages 149–156, 2008.

[227] Robert A Scheidt, Michael A Conditt, Emanuele L Secco, and Ferdinando A Mussa-Ivaldi. Interaction of visual and proprioceptive feedback during adaptation of human reaching movements. *Journal of neurophysiology*, 93(6):3200–3213, 2005.

[228] Christine Tong, Daniel M Wolpert, and J Randall Flanagan. Kinematics and dynamics are not represented independently in motor working memory: evidence from an interference study. *Journal of Neuroscience*, 22(3):1108–1113, 2002.

[229] John W Krakauer, Maria-Felice Ghilardi, and Claude Ghez. Independent learning of internal models for kinematic and dynamic control of reaching. *Nature neuroscience*, 2(11):1026–1031, 1999.

[230] David E Meyer, Richard A Abrams, Sylvan Kornblum, Charles E Wright, and JE Keith Smith. Optimality in human motor performance: ideal control of rapid aimed movements. *Psychological review*, 95(3):340, 1988.

[231] Richard Magill and David I Anderson. *Motor learning and control*. McGraw-Hill Publishing New York, 2010.

[232] Alexandre Costa Henriques and Ingrid Winkler. The advancement of virtual reality in automotive market research: Challenges and opportunities. *Applied Sciences*, 11(24):11610, 2021.

[233] Carolee Winstein, Rebecca Lewthwaite, Sarah R Blanton, Lois B Wolf, and Laurie Wishart. Infusing motor learning research into neurorehabilitation practice: a historical perspective with case exemplar from the accelerated skill acquisition program. *Journal of neurologic physical therapy: JNPT*, 38(3):190, 2014.

[234] Irene Valori, Phoebe E McKenna-Plumley, Rena Bayramova, Claudio Zandonella Callegher, Gianmarco Altoè, and Teresa Farroni. Proprioceptive accuracy in immersive virtual reality: A developmental perspective. *PloS one*, 15(1):e0222253, 2020.

[235] Gerard G Fluet and Judith E Deutsch. Virtual reality for sensorimotor rehabilitation post-stroke: the promise and current state of the field. *Current physical medicine and rehabilitation reports*, 1:9–20, 2013.

[236] John W Krakauer, Zachary M Pine, Maria-Felice Ghilardi, and Claude Ghez. Learning of visuomotor transformations for vectorial planning of reaching trajectories. *Journal of neuroscience*, 20(23):8916–8924, 2000.

[237] Matthis Synofzik, Axel Lindner, and Peter Thier. The cerebellum updates predictions about the visual consequences of one's behavior. *Current Biology*, 18(11):814–818, 2008.

[238] Daniel M Wolpert, R Chris Miall, and Mitsuo Kawato. Internal models in the cerebellum. *Trends in cognitive sciences*, 2(9):338–347, 1998.

[239] Denise YP Henriques and Erin K Cressman. Visuomotor adaptation and proprioceptive recalibration. *Journal of motor behavior*, 44(6):435–444, 2012.

[240] Danielle Salomonczyk, Erin K Cressman, and Denise YP Henriques. The role of the cross-sensory error signal in visuomotor adaptation. *Experimental Brain Research*, 228:313–325, 2013.

[241] Erin K Cressman and Denise YP Henriques. Reach adaptation and proprioceptive recalibration following exposure to misaligned sensory input. *Journal of neurophysiology*, 103(4):1888–1895, 2010.

[242] Denise YP Henriques, Filipp Filippopulos, Andreas Straube, and Thomas Eggert. The cerebellum is not necessary for visually driven recalibration of hand proprioception. *Neuropsychologia*, 64:195–204, 2014.

[243] Basel Zbib, Denise YP Henriques, and Erin K Cressman. Proprioceptive recalibration arises slowly compared to reach adaptation. *Experimental Brain Research*, 234:2201–2213, 2016.

[244] Jennifer E Ruttle, Erin K Cressman, Bernard Marius 't Hart, and Denise YP Henriques. Time course of reach adaptation and proprioceptive recalibration during visuomotor learning. *PLoS One*, 11(10):e0163695, 2016.

[245] Ahmed A Mostafa, Bernard Marius 't Hart, and Denise YP Henriques. Motor learning without moving: proprioceptive and predictive hand localization after passive visuoproprioceptive discrepancy training. *PLoS One*, 14(8):e0221861, 2019.

[246] Brendan D Cameron, Ian M Franks, J Timothy Inglis, and Romeo Chua. The adaptability of self-action perception and movement control when the limb is passively versus actively moved. *Consciousness and Cognition*, 21(1):4–17, 2012.

[247] Jason Stanley and John W Krakauer. Motor skill depends on knowledge of facts. *Frontiers in human neuroscience*, 7:503, 2013.

[248] Waqas Sultani, Chen Chen, and Mubarak Shah. Real-world anomaly detection in surveillance videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6479–6488, 2018.

[249] Ritesh K Malaiya, Donghwoon Kwon, Sang C Suh, Hyunjoo Kim, Ikkyun Kim, and Jinoh Kim. An empirical evaluation of deep learning for network anomaly detection. *IEEE Access*, 7:140806–140817, 2019.

[250] Sharmin Aktar and Abdullah Yasin Nur. Towards ddos attack detection using deep learning approach. *Computers & Security*, 129:103251, 2023.

[251] Franco Van Wyk, Yiyang Wang, Anahita Khojandi, and Neda Masoud. Real-time sensor anomaly detection and identification in automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):1264–1276, 2019.

[252] Oleksandr I Provotar, Yaroslav M Linder, and Maksym M Veres. Unsupervised anomaly detection in time series using lstm-based autoencoders. In *2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT)*, pages 513–517. IEEE, 2019.

[253] Alex Graves and Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.

[254] Sucheta Chauhan and Lovekesh Vig. Anomaly detection in ecg time signals via deep long short-term memory networks. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pages 1–7. IEEE, 2015.

[255] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[256] Kun Yang, Junjie Zhang, Yang Xu, and Jonathan Chao. Ddos attacks detection with autoencoder. In *NOMS 2020-2020 IEEE/IFIP network operations and management symposium*, pages 1–9. IEEE, 2020.

[257] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Computing Surveys (CSUR)*, 39(1):3–es, 2007.

[258] Hyunseung Choi, Mintae Kim, Gyubok Lee, and Wooju Kim. Unsupervised learning approach for network intrusion detection system using autoencoders. *The Journal of Supercomputing*, 75:5597–5621, 2019.

[259] Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. Network anomaly detection using lstm based autoencoder. In *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, pages 37–45, 2020.

[260] Rick Kjeldsen and Jacob Hartman. Design issues for vision-based computer interaction systems. In *Proceedings of the 2001 Workshop on Perceptive User Interfaces*, PUI '01, page 1–8, New York, NY, USA, 2001. Association for Computing Machinery.

[261] Yasmin Felberbaum and Joel Lanir. Step by step: investigating foot gesture interaction. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 306–307, 2016.

[262] Christopher R Austin, Barrett Ens, Kadek Ananta Satriadi, and Bernhard Jenny. Elicitation study investigating hand and foot gesture interaction for immersive maps in augmented reality. *Cartography and Geographic Information Science*, 47(3):214–228, 2020.

[263] Zhanming Chen, Huawei Tu, and Huiyue Wu. User-defined foot gestures for eyes-free interaction in smart shower rooms. *International Journal of Human–Computer Interaction*, pages 1–23, 2022.

[264] Rebecca Fiebrink, Dan Trueman, and Perry R Cook. A meta-instrument for interactive, on-the-fly machine learning. In *NIME*, pages 280–285, 2009.

[265] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

[266] Ruoyu Zhi. A drift eliminated attitude & position estimation algorithm in 3d. 2016.

[267] L Sher. Personal inertial navigation system (pins). DARPA, 1996.

[268] Eric Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer graphics and applications*, 25(6):38–46, 2005.

[269] Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254, 1985.

[270] Nurettin Çağrı Kılıboz and Uğur Güdükbay. A hand gesture recognition technique for human–computer interaction. *Journal of Visual Communication and Image Representation*, 28:97–104, 2015.

[271] Kenrick Kin, Björn Hartmann, Tony DeRose, and Maneesh Agrawala. Proton: multitouch gestures as regular expressions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2885–2894, 2012.

[272] Kenrick Kin, Björn Hartmann, Tony DeRose, and Maneesh Agrawala. Proton++ a customizable declarative multitouch framework. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 477–486, 2012.

[273] Alessandro Carcangiu, Lucio Davide Spano, Giorgio Fumera, and Fabio Roli. Deictic: A compositional and declarative gesture description based on hidden markov models. *International Journal of Human-Computer Studies*, 122:113–132, 2019.

[274] Lucio Davide Spano, Antonio Cisternino, Fabio Paternò, and Gianni Fenu. Gestit: a declarative and compositional framework for multiplatform gesture definition. In *Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems*, pages 187–196, 2013.

[275] Alessandro Carcangiu and Lucio Davide Spano. G-gene: A gene alignment method for online partial stroke gestures recognition. *Proceedings of the ACM on Human-Computer Interaction*, 2(EICS):1–17, 2018.

[276] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[277] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[278] Pengyu Hong, Matthew Turk, and Thomas S Huang. Gesture modeling and recognition using finite state machines. In *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, pages 410–415. IEEE, 2000.

[279] Jacob O Wobbrock, Leah Findlater, Darren Gergle, and James J Higgins. The aligned rank transform for nonparametric factorial analyses using only anova procedures. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 143–146, 2011.

[280] Google. Use wrist gestures on Wear. https://developer.android.com/training/wearables/user-input/wrist-gestures, 2022. Online; accessed Feb 2022.

[281] John Salatas. HandsFree Wear: A wrist gesture input method for Android based smartwatches. https://jsalatas.ictpro.gr/handsfree-wear-a-wrist-gesture-input-method-for-android-based-smartwatches/, 2022. Online; accessed Feb 2022.

[282] Radu-Daniel Vatavu, Daniel Vogel, Géry Casiez, and Laurent Grisoni. Estimating the perceived difficulty of pen gestures. In *IFIP Conference on Human-Computer Interaction*, pages 89–106. Springer, 2011.

[283] Juan David Hincapié-Ramos, Xiang Guo, Paymahn Moghadasian, and Pourang Irani. Consumed endurance: a metric to quantify arm fatigue of mid-air interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1063–1072, 2014.

[284] Md Aashikur Rahman Azim, Adil Rahman, and Seongkook Heo. Over-the-shoulder training between redundant wearable sensors for unified gesture interactions. In *Adjunct Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, pages 1–3, 2022.

[285] Yizheng Gu, Chun Yu, Zhipeng Li, Weiqi Li, Shuchang Xu, Xiaoying Wei, and Yuanchun Shi. Accurate and Low-Latency Sensing of Touch Contact on Any Surface with Finger-Worn IMU Sensor. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, pages 1059–1070, New Orleans LA USA, October 2019. ACM.

[286] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[287] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[288] Brian Kenji Iwana and Seiichi Uchida. An empirical survey of data augmentation for time series classification with neural networks. *Plos one*, 16(7):e0254841, 2021.

[289] Facebook's AI Research lab. PyTorch Manual: The developer's perspective. https://pytorch.org/, 2022. Online; accessed Aug 2022.

[290] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[291] Nils Y Hammerla and Thomas Plötz. Let's (not) stick together: pairwise similarity biases cross-validation in activity recognition. In *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing*, pages 1041–1051, 2015.

[292] Chulhong Min, Seungwoo Kang, Chungkuk Yoo, Jeehoon Cha, Sangwon Choi, Younghan Oh, and Junehwa Song. Exploring current practices for battery use and management of smartwatches. In *Proceedings of the 2015 ACM international symposium on wearable computers*, pages 11–18, 2015.

[293] Tom Ouyang and Yang Li. Bootstrapping personal gesture shortcuts with the wisdom of the crowd and handwriting recognition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2895–2904, 2012.

[294] Antti Salovaara, Antti Oulasvirta, and Giulio Jacucci. Evaluation of prototypes and the problem of possible futures. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 2064–2077, 2017.

[295] Antti Oulasvirta. Feature when users" do" the ubicomp. *Interactions*, 15(2):6–9, 2008.

[296] Paul Dourish. *Where the action is: the foundations of embodied interaction*. MIT press, 2004.

[297] Matthew Chalmers and Areti Galani. Seamful interweaving: heterogeneity in the theory and design of interactive systems. In *Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques*, pages 243–252, 2004.

[298] James J Gibson. Adaptation, after-effect and contrast in the perception of curved lines. *Journal of experimental psychology*, 16(1):1, 1933.

[299] Luv Kohli. *Redirected touching*. PhD thesis, The University of North Carolina at Chapel Hill, 2013.

[300] Parastoo Abtahi and Sean Follmer. Visuo-haptic illusions for improving the perceived performance of shape displays. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2018.

[301] Johann Wentzel, Greg d'Eon, and Daniel Vogel. Improving virtual reality ergonomics through reach-bounded non-linear input amplification. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2020.

[302] Eric J Gonzalez and Sean Follmer. Investigating the detection of bimanual haptic retargeting in virtual reality. In *Proceedings of the 25th ACM Symposium on Virtual Reality Software and Technology*, pages 1–5, 2019.

[303] André Zenner and Antonio Krüger. Estimating detection thresholds for desktop-scale hand redirection in virtual reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 47–55. IEEE, 2019.

[304] Shaghayegh Esmaeili, Brett Benda, and Eric D Ragan. Detection of scaled hand interactions in virtual reality: The effects of motion direction and task complexity. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 453–462. IEEE, 2020.

[305] Brett Benda, Shaghayegh Esmaeili, and Eric D Ragan. Determining detection thresholds for fixed positional offsets for virtual hand remapping in virtual reality. In *2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 269–278. IEEE, 2020.

[306] Luv Kohli, Mary C Whitton, and Frederick P Brooks. Redirected touching: The effect of warping space on task performance. In *2012 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 105–112. IEEE, 2012.

[307] Dustin T Han, Mohamed Suhail, and Eric D Ragan. Evaluating remapped physical reach for hand interactions with passive haptics in virtual reality. *IEEE transactions on visualization and computer graphics*, 24(4):1467–1476, 2018.

[308] Xuanhui Yang, Yan Zhang, and Xubo Yang. Redirected placement: Evaluating the redirection of passive props during reach-to-place in virtual reality. In *Proceedings of the 29th ACM Symposium on Virtual Reality Software and Technology*, pages 1–11, 2023.

[309] Robert Sessions Woodworth. Accuracy of voluntary movement. *The Psychological Review: Monograph Supplements*, 3(3):i, 1899.

[310] Jennifer E Ruttle, Bernard Marius 't Hart, and Denise YP Henriques. The fast contribution of visual-proprioceptive discrepancy to reach aftereffects and proprioceptive recalibration. *PLoS One*, 13(7):e0200621, 2018.

[311] Digby Elliott, Steve Hansen, Lawrence EM Grierson, James Lyons, Simon J Bennett, and Spencer J Hayes. Goal-directed aiming: two components but multiple processes. *Psychological bulletin*, 136(6):1023, 2010.

[312] James Lyons, Steve Hansen, Suzanne Hurding, and Digby Elliott. Optimizing rapid aiming behaviour: Movement kinematics depend on the cost of corrective modifications. *Experimental Brain Research*, 174:95–100, 2006.

[313] Logan D Clark and Sara L Riggs. Movement strategies in virtual reality: The influence of effort costs and target depth. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 64, pages 1600–1604. SAGE Publications Sage CA: Los Angeles, CA, 2020.

[314] Logan D Clark, Mohamad El Iskandarani, and Sara L Riggs. The effect of movement direction, hand dominance, and hemispace on reaching movement kinematics in virtual reality. In *Proceedings of the 2023 CHI conference on human factors in computing systems*, pages 1–18, 2023.

[315] Charalambos Papaxanthis, Thierry Pozzo, and Marco Schieppati. Trajectories of arm pointing movements on the sagittal plane vary with both direction and speed. *Experimental brain research*, 148:498–503, 2003.

[316] Unnikrishnan Radhakrishnan, Konstantinos Koumaditis, and Francesco Chinello. A systematic review of immersive virtual reality for industrial skills training. *Behaviour & Information Technology*, 40(12):1310–1339, 2021.

[317] Joseph P Salisbury, Ted M Aronson, and Tony J Simon. At-home self-administration of an immersive virtual reality therapeutic game for post-stroke upper limb rehabilitation. In *Extended abstracts of the 2020 annual symposium on computer-human interaction in play*, pages 114–121, 2020.

[318] Robert Fisher, Simon Perkins, A Walker, and E Wolfart. Gaussian smoothing. *Hypermedia Image Processing Reference*, 2003.

[319] Richard A Abrams and Jay Pratt. Rapid aimed limb movements: Differential effects of practice on component submovements. *Journal of motor behavior*, 25(4):288–298, 1993.

[320] RUMELHART DE. Learning representations by back-propagation errors. *nature*, 323:533–536, 1986.

[321] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *Machine learning for data science handbook: data mining and knowledge discovery handbook*, pages 353–374, 2023.

[322] Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 665–674, 2017.

[323] Jayawant N Mandrekar. Receiver operating characteristic curve in diagnostic test assessment. *Journal of Thoracic Oncology*, 5(9):1315–1316, 2010.

[324] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[325] Gerd Bruder, Paul Lubos, and Frank Steinicke. Cognitive resource demands of redirected walking. *IEEE transactions on visualization and computer graphics*, 21(4):539–544, 2015.

[326] Timofey Grechkin, Jerald Thomas, Mahdi Azmandian, Mark Bolas, and Evan Suma. Revisiting detection thresholds for redirected walking: Combining translation and curvature gains. In *Proceedings of the ACM symposium on applied perception*, pages 113–120, 2016.

[327] Hyunsu Cho, Sunwoo Lee, Wonsuk Choi, and Dong Hoon Lee. Recovering yaw rate from signal injection attack to protect rv's direction. In *International Conference on Information Security Applications*, pages 171–184. Springer, 2022.

[328] Zhan Tu, Fan Fei, Matthew Eagon, Dongyan Xu, and Xinyan Deng. Flight recovery of mavs with compromised imu. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3638–3644. IEEE, 2019.

[329] Sivakumar Balasubramanian, Alejandro Melendez-Calderon, and Etienne Burdet. A robust and sensitive metric for quantifying movement smoothness. *IEEE transactions on biomedical engineering*, 59(8):2126–2136, 2011.

[330] Sushmita Mitra and Tinku Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3):311–324, 2007.

[331] Hong Cheng, Lu Yang, and Zicheng Liu. Survey on 3d hand gesture recognition. *IEEE transactions on circuits and systems for video technology*, 26(9):1659–1673, 2015.

[332] Rihem Mahmoud, Selma Belgacem, and Mohamed Nazih Omri. Deep signature-based isolated and large scale continuous gesture recognition approach. *Journal of King Saud University-Computer and Information Sciences*, 2020.

[333] Gonzalo Bailador, Daniel Roggen, Gerhard Tröster, and Gracián Triviño. Real time gesture recognition using continuous time recurrent neural networks. In *BodyNets*, page 15, 2007.

[334] Eugene M Taranta II, Amirreza Samiei, Mehran Maghoumi, Pooya Khaloo, Corey R Pittman, and Joseph J LaViola Jr. Jackknife: A reliable recognizer with few samples and many modalities. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 5850–5861, 2017.

[335] Sait Celebi, Ali Selman Aydin, Talha Tarik Temiz, and Tarik Arici. Gesture recognition using skeleton data with weighted dynamic time warping. In *VISAPP (1)*, pages 620–625, 2013.

[336] Saša Bodiroža, Guillaume Doisy, and Verena Vanessa Hafner. Position-invariant, real-time gesture recognition based on dynamic time warping. In *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 87–88. IEEE, 2013.

[337] Michael Nebeling, David Ott, and Moira C Norrie. Kinect analysis: a system for recording, analysing and sharing multimodal interaction elicitation studies. In *Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pages 142–151, 2015.

[338] Sharad Vikram, Lei Li, and Stuart Russell. Writing and sketching in the air, recognizing and controlling on the fly. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 1179–1184. 2013.

[339] Ada Wai-Chee Fu, Eamonn Keogh, Leo Yung Hang Lau, Chotirat Ann Ratanamahatana, and Raymond Chi-Wing Wong. Scaling and time warping in time series querying. *The VLDB Journal*, 17(4):899–921, 2008.

[340] Jonathan Wu, Janusz Konrad, and Prakash Ishwar. Dynamic time warping for gesture-based user identification and authentication with kinect. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2371–2375. IEEE, 2013.

[341] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data mining and knowledge discovery*, 31(3):606–660, 2017.

[342] Rafael Giusti and Gustavo EAPA Batista. An empirical comparison of dissimilarity measures for time series classification. In *2013 Brazilian Conference on Intelligent Systems*, pages 82–88. IEEE, 2013.

[343] Mauricio Cirelli and Ricardo Nakamura. A survey on multi-touch gesture recognition and multi-touch frameworks. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, pages 35–44, 2014.

[344] Zachary C. Lipton. The mythos of model interpretability. *Commun. ACM*, 61(10):36–43, sep 2018.

[345] Tim Dittmar, Claudia Krull, and Graham Horton. A new approach for touch gesture recognition: Conversive hidden non-markovian models. *Journal of Computational Science*, 10:66–76, 2015.

[346] Tevfik Metin Sezgin and Randall Davis. Hmm-based efficient sketch recognition. In *Proceedings of the 10th international conference on Intelligent user interfaces*, pages 281–283, 2005.

[347] Pradeep Natarajan, Vivek Kumar Singh, and Ram Nevatia. Learning 3d action models from a few 2d videos for view invariant action recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 20006–2013. IEEE, 2010.

[348] Pradeep Natarajan and Ramakant Nevatia. Online, real-time tracking and recognition of human actions. In *2008 IEEE Workshop on Motion and video Computing*, pages 1–8. IEEE, 2008.

[349] Benjamin Poppinga, Alireza Sahami Shirazi, Niels Henze, Wilko Heuten, and Susanne Boll. Understanding shortcut gestures on mobile touch devices. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*, pages 173–182, 2014.

[350] A Chris Long, James A Landay, and Lawrence A Rowe. " those look similar!" issues in automating gesture design advice. In *Proceedings of the 2001 workshop on Perceptive user interfaces*, pages 1–5, 2001.

[351] Björn Hartmann, Scott R. Klemmer, Michael Bernstein, Leith Abdulla, Brandon Burr, Avi Robinson-Mosher, and Jennifer Gee. Reflective physical prototyping through integrated design, test, and analysis. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, UIST '06, page 299–308, New York, NY, USA, 2006. Association for Computing Machinery.

[352] Beryl Plimmer, Rachel Blagojevic, Samuel Hsiao-Heng Chang, Paul Schmieder, and Jacky Shunjie Zhen. Rata: codeless generation of gesture recognizers. In *The 26th BCS Conference on Human Computer Interaction 26*, pages 137–146, 2012.

[353] Hao Lü, James A Fogarty, and Yang Li. Gesture script: Recognizing gestures and their structure using rendering scripts and interactively trained parts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1685–1694, 2014.

[354] Beat Signer, Ueli Kurmann, and Moira Norrie. igesture: a general gesture recognition framework. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 954–958. IEEE, 2007.

[355] Hao Lü and Yang Li. Gesture coder: a tool for programming multi-touch gestures by demonstration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2875–2884, 2012.

[356] Anind K Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. a cappella: programming by demonstration of context-aware applications. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 33–40, 2004.

[357] Hao Lü and Yang Li. Gesture studio: authoring multi-touch interactions through demonstration and declaration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 257–266, 2013.

[358] Shahedul Huq Khandkar, SM Sohan, Jonathan Sillito, and Frank Maurer. Tool support for testing complex multi-touch gestures. In *ACM International Conference on Interactive Tabletops and Surfaces*, pages 59–68, 2010.

[359] Nathan Magrofuoco, Paolo Roselli, Jean Vanderdonckt, Jorge Luis Pérez-Medina, and Radu-Daniel Vatavu. Gestman: a cloud-based tool for stroke-gesture datasets. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pages 1–6, 2019.

[360] Jacob O Wobbrock, Andrew D Wilson, and Yang Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 159–168, 2007.

[361] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O Wobbrock. Gestures as point clouds: a $p recognizer for user interface prototypes. In *Proceedings of the 14th ACM international conference on Multimodal interaction*, pages 273–280, 2012.

[362] Nathan Magrofuoco and Jean Vanderdonckt. Gelicit: a cloud platform for distributed gesture elicitation studies. *Proceedings of the ACM on Human-Computer Interaction*, 3(EICS):1–41, 2019.

[363] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O Wobbrock. Relative accuracy measures for stroke gestures. In *Proceedings of the 15th ACM on International conference on multimodal interaction*, pages 279–286, 2013.

[364] Songyang Lao, Xiangan Heng, Guohua Zhang, Yunxiang Ling, and Peng Wang. A gestural interaction design model for multi-touch displays. *People and Computers XXIII Celebrating People and Technology*, pages 440–446, 2009.

[365] Muhammad Haroon and Malik Sikandar Hayat Khiyal. A gesture recognition design toolkit everyday gesture library. *International Journal of Computer Applications*, 121(14), 2015.

[366] Birgit Bomsdorf, Rainer Blum, and Daniel Künkel. Towards progesture, a tool supporting early prototyping of 3d-gesture interaction. In *3D Printing: Breakthroughs in Research and Practice*, pages 396–413. IGI Global, 2017.

[367] Fabrizio Milazzo, Vito Gentile, Antonio Gentile, and Salvatore Sorce. Kind-dama: A modular middleware for kinect-like device data management. *Software: Practice and Experience*, 48(1):141–160, 2018.

[368] Mehmet Aydın Baytaş, Yücel Yemez, and Oğuzhan Özcan. Hotspotizer: end-user authoring of mid-air gestural interactions. In *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational*, pages 677–686, 2014.

[369] C Henrique and Q Forster. Design of gesture vocabularies through analysis of recognizer performance in gesture space. In *Seventh International Conference on Intelligent Systems Design and Applications (ISDA 2007)*, pages 641–646. IEEE, 2007.

[370] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O Wobbrock. Gesture heatmaps: Understanding gesture performance with colorful visualizations. In *Proceedings of the 16th International Conference on Multimodal Interaction*, pages 172–179, 2014.

[371] Lisa Anthony, Radu-Daniel Vatavu, and Jacob O Wobbrock. Understanding the consistency of users' pen and finger stroke gesture articulation. In *Proceedings of Graphics Interface 2013*, pages 87–94. Citeseer, 2013.

[372] Huy Viet Le, Sven Mayer, Benedict Steuerlein, and Niels Henze. Investigating unintended inputs for one-handed touch interaction beyond the touchscreen. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 1–14, 2019.

[373] Euan Freeman, Gareth Griffiths, and Stephen A Brewster. Rhythmic micro-gestures: Discreet interaction on-the-go. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pages 115–119, 2017.

[374] Alan Bränzel, Christian Holz, Daniel Hoffmann, Dominik Schmidt, Marius Knaust, Patrick Lühne, René Meusel, Stephan Richter, and Patrick Baudisch. Gravityspace: tracking users and their poses in a smart room using a pressure-sensing floor. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 725–734, 2013.

[375] Ling Bao and Stephen S Intille. Activity recognition from user-annotated acceleration data. In *International conference on pervasive computing*, pages 1–17. Springer, 2004.

[376] Eduardo Velloso, Dominik Schmidt, Jason Alexander, Hans Gellersen, and Andreas Bulling. The feet in human–computer interaction: A survey of foot-based interaction. *ACM Computing Surveys (CSUR)*, 48(2):1–35, 2015.

[377] Lars Erik Holmquist, Friedemann Mattern, Bernt Schiele, Petteri Alahuhta, Michael Beigl, and Hans-W Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *international conference on Ubiquitous Computing*, pages 116–122. Springer, 2001.

[378] Joseph A Paradiso, Stacy J Morris, Ari Y Benbasat, and Erik Asmussen. Interactive therapy with instrumented footwear. In *CHI'04 Extended Abstracts on Human Factors in Computing Systems*, pages 1341–1343, 2004.

[379] Toshiki Iso and Kenichi Yamazaki. Gait analyzer based on a cell phone with a single three-axis accelerometer. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*, pages 141–144, 2006.

[380] Scott E Hudson, Chris Harrison, Beverly L Harrison, and Anthony LaMarca. Whack gestures: inexact and inattentive interaction with mobile devices. In *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*, pages 109–112, 2010.

[381] Rolf Inge Godøy, Minho Song, Kristian Nymoen, Mari Romarheim Haugen, and Alexander Refsum Jensenius. Exploring sound-motion similarity in musical experience. *Journal of New Music Research*, 45(3):210–222, 2016.

[382] Daisuke Wakabayashi. Apple watch users discover another way to go 'hands free'. *The Wall Street Journal*, December 25 2015.

[383] Ondrej Polacek, Thomas Grill, and Manfred Tscheligi. Nosetapping: What else can you do with your nose? In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, MUM '13, New York, NY, USA, 2013. Association for Computing Machinery.

[384] Juyoung Lee, Shaurye Aggarwal, Jason Wu, Thad Starner, and Woontack Woo. Selfsync: Exploring self-synchronous body-based hotword gestures for initiating interaction. In *Proceedings of the 23rd International Symposium on Wearable Computers*, ISWC '19, page 123–128, New York, NY, USA, 2019. Association for Computing Machinery.

[385] Augusto Esteves, Eduardo Velloso, Andreas Bulling, and Hans Gellersen. Orbits: Gaze interaction for smart watches using smooth pursuit eye movements. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15, page 457–466, New York, NY, USA, 2015. Association for Computing Machinery.

[386] Adam Zarek, Daniel Wigdor, and Karan Singh. Snout: One-handed use of capacitive touch devices. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, page 140–147, New York, NY, USA, 2012. Association for Computing Machinery.

[387] Julie Rico and Stephen Brewster. Usable gestures for mobile interfaces: evaluating social acceptability. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 887–896, 2010.

[388] Kurt Bedell, Adam H Buchaklian, and Stanley Perlman. Efficacy of an automated multiple emitter whole-room ultraviolet-c disinfection system against coronaviruses mhv and mers-cov. *infection control & hospital epidemiology*, 37(5):598–599, 2016.

[389] Marion Koelle, Swamy Ananthanarayan, and Susanne Boll. Social acceptability in hci: A survey of methods, measures, and design strategies. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–19, 2020.

[390] Mahesh Pal. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222, 2005.

[391] Irina Rish et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[392] David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. *Logistic regression*. Springer, 2002.

[393] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[394] Romain Tavenard, Johann Faouzi, Gilles Vandewiele, Felix Divo, Guillaume Androz, Chester Holtz, Marie Payne, Roman Yurchak, Marc Rußwurm, Kushal Kolar, and Eli Woods. Tslearn, a machine learning toolkit for time series data. *Journal of Machine Learning Research*, 21(118):1–6, 2020.

[395] John Elwell. Inertial navigation for the urban warrior. In *Digitization of the Battlespace IV*, volume 3709, pages 196–204. International Society for Optics and Photonics, 1999.

[396] Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '85, page 245–254, New York, NY, USA, 1985. Association for Computing Machinery.

[397] Qifan Zhou, Hai Zhang, Zahra Lari, Zhenbo Liu, and Naser El-Sheimy. Design and implementation of foot-mounted inertial sensor based wearable electronic device for game play application. *Sensors*, 16(10):1752, 2016.

[398] Janice Kaye Loudon, Marcie Swift, and Stephania Bell. *The clinical orthopedic assessment guide*. Human Kinetics, 2008.

[399] Sandra J Shultz, Peggy A Houglum, and David H Perrin. *Examination of musculoskeletal injuries*. Human Kinetics, 2015.

[400] Abdul Razak, Abdul Hadi, Aladin Zayegh, Rezaul K Begg, and Yufridin Wahab. Foot plantar pressure measurement system: A review. *Sensors*, 12(7):9884–9912, 2012.

[401] Thomas Augsten, Konstantin Kaefer, René Meusel, Caroline Fetzer, Dorian Kanitz, Thomas Stoff, Torsten Becker, Christian Holz, and Patrick Baudisch. Multitoe: high-precision interaction with back-projected floors based on high-resolution multi-touch input. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 209–218. ACM, 2010.

[402] Sunjun Kim and Geehyuk Lee. Tapboard 2: Simple and effective touchpad-like interaction on a multi-touch surface keyboard. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 5163–5168. ACM, 2016.

[403] Henry Sweet. *The history of language*. Dent, 1900.

[404] Benjamin Hatscher, Maria Luz, and Christian Hansen. Foot interaction concepts to support radiological interventions. *i-com*, 17(1):3–13, 2018.

[405] Ambreen Zaman, Lars Reisig, Anke Verena Reinschluessel, Huseyin Bektas, Dirk Weyhe, Marc Herrlich, Tanja Döring, and Rainer Malaka. An interactive-shoe for surgeons: Hand-free interaction with medical 2d data. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, page LBW633. ACM, 2018.

[406] Yanpei Huang, Etienne Burdet, Lin Cao, Phuoc Thien Phan, Anthony Meng Huat Tiong, Pai Zheng, and Soo Jay Phee. Performance evaluation of a foot interface to operate a robot arm. *IEEE Robotics and Automation Letters*, 4(4):3302–3309, 2019.

[407] Brandon William Rudolph. Foot-controlled supernumerary robotic arm: Foot interfaces and human abilities. 2019.

[408] Jason Alexander, Teng Han, William Judd, Pourang Irani, and Sriram Subramanian. Putting your best foot forward: investigating real-world mappings for foot-based gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1229–1238. ACM, 2012.

[409] Sebastian OH Madgwick, Andrew JL Harrison, and Ravi Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *2011 IEEE international conference on rehabilitation robotics*, pages 1–7. IEEE, 2011.

[410] Saurabh Godha and Gerard Lachapelle. Foot mounted inertial system for pedestrian navigation. *Measurement Science and Technology*, 19(7):075202, 2008.

[411] Byron Lahey, Audrey Girouard, Winslow Burleson, and Roel Vertegaal. Paperphone: understanding the use of bend gestures in mobile devices with flexible electronic paper displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1303–1312. ACM, 2011.

[412] Jacob O Wobbrock, Htet Htet Aung, Brandon Rothrock, and Brad A Myers. Maximizing the guessability of symbolic input. In *CHI'05 extended abstracts on Human Factors in Computing Systems*, pages 1869–1872. ACM, 2005.

[413] HM Chai. Applications of kinesiology–gait during ambulation. *Web: http://www.pt.ntu.edu.tw/hmchai/Kines04/KINapplication/Gait.htm*, 2004.

[414] Mark McClusky. The nike experiment: how the shoe giant unleashed the power of personal metrics. *Wired*, 17(07), 2009.

[415] Scott R Klemmer, Anoop K Sinha, Jack Chen, James A Landay, Nadeem Aboobaker, and Annie Wang. Suede: a wizard of oz prototyping tool for speech user interfaces. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 1–10, 2000.

[416] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[417] Bill Waggener, William N Waggener, and William M Waggener. *Pulse code modulation techniques*. Springer Science & Business Media, 1995.

[418] Wikipedia. Inertial navigation system. https://en.wikipedia.org/wiki/Inertial_navigation_system, 2022. Online; accessed Feb 2022.

[419] Maximilian Speicher and Michael Nebeling. Gesturewiz: A human-powered gesture design environment for user interface prototypes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–11, 2018.

[420] Google Inc. Android Developer Manual: The developer's perspective. https://developer.android.com/reference/android/hardware/Sensor#TYPE_GAME_ROTATION_VECTOR, 2022. Online; accessed Aug 2022.

[421] Javier Hernandez, Yin Li, James M Rehg, and Rosalind W Picard. Bioglass: Physiological parameter estimation using a head-mounted wearable device. In *2014 4th International Conference on Wireless Mobile Communication and Healthcare-Transforming Healthcare Through Innovations in Mobile and Wireless Technologies (MOBIHEALTH)*, pages 55–58. IEEE, 2014.

[422] Dimitrios Raptis, Jesper Kjeldskov, and Mikael B Skov. Continuity in multi-device interaction: an online study. In *Proceedings of the 9th nordic conference on human-computer interaction*, pages 1–10, 2016.

[423] Steven Hoober. How do users really hold mobile devices. *Uxmatters (http://www. uxmatter. com). Published: Feburary*, 18:2327–4662, 2013.

[424] Aleksandr Ometov, Viktoriia Shubina, Lucie Klus, Justyna Skibińska, Salwa Saafi, Pavel Pascacio, Laura Flueratoru, Darwin Quezada Gaibor, Nadezhda Chukhno, Olga Chukhno, et al. A survey on wearable technology: History, state-of-the-art and current challenges. *Computer Networks*, 193:108074, 2021.

[425] Russel Pears, Jacqui Finlay, and Andy M Connor. Synthetic minority over-sampling technique (smote) for predicting software build outcomes. *arXiv preprint arXiv:1407.2330*, 2014.

[426] Jun Rekimoto. Pick-and-drop: a direct manipulation technique for multiple computer environments. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 31–39, 1997.

[427] Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, and Franz J Király. sktime: A unified interface for machine learning with time series. *arXiv preprint arXiv:1909.07872*, 2019.

[428] Dominik Schmidt, Fadi Chehimi, Enrico Rukzio, and Hans Gellersen. Phonetouch: a technique for direct phone interaction on surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 13–16, 2010.

[429] Robert Hardy and Enrico Rukzio. Touch & interact: touch-based interaction of mobile phones with displays. In *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services*, pages 245–254, 2008.

[430] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[431] Mark Weiser. The computer for the 21st century. *ACM SIGMOBILE mobile computing and communications review*, 3(3):3–11, 1999.

[432] Tran Huy Vu, Archan Misra, Quentin Roy, Kenny Choo Tsu Wei, and Youngki Lee. Smartwatch-based early gesture detection 8 trajectory tracking for interactive gesture-driven applications. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(1):1–27, 2018.

[433] Franca Alexandra Rupprecht, Achim Ebert, Andreas Schneider, and Bernd Hamann. Virtual reality meets smartwatch: Intuitive, natural, and multi-modal interaction. In *Proceedings of the 2017 chi conference extended abstracts on human factors in computing systems*, pages 2884–2890, 2017.

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

[435] Danial Moazen, Seyed A Sajjadi, and Ani Nahapetian. AirDraw: Leveraging smart watch motion sensors for mobile human computer interactions. In *2016 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 442–446, January 2016. ISSN: 2331-9860.

[436] David Mace, Wei Gao, and Ayse Coskun. Accelerometer-based hand gesture recognition using feature weighted naïve bayesian classifiers and dynamic time warping. In *Proceedings of the companion publication of the 2013 international conference on Intelligent user interfaces companion - IUI '13 Companion*, page 83, Santa Monica, California, USA, 2013. ACM Press.

[437] Lorenzo Porzi, Stefano Messelodi, Carla Mara Modena, and Elisa Ricci. 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 - IMMPD '13*, pages 19–24, Barcelona, Spain, 2013. ACM Press.

[438] Yannick Bernaerts, Matthias Druwé, Sebastiaan Steensels, Jo Vermeulen, and Johannes Schöning. The office smartwatch: development and design of a smartwatch app to digitally augment interactions in an office environment. In *Proceedings of the 2014 companion publication on Designing interactive systems - DIS Companion '14*, pages 41–44, Vancouver, BC, Canada, 2014. ACM Press.

[439] Hui-Shyong Yeo, Juyoung Lee, Hyung-il Kim, Aakar Gupta, Andrea Bianchi, Daniel Vogel, Hideki Koike, Woontack Woo, and Aaron Quigley. WRIST: Watch-Ring Interaction and Sensing Technique for Wrist Gestures and Macro-Micro Pointing. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 1–15, Taipei Taiwan, October 2019. ACM.

[440] Jun Gong, Aakar Gupta, and Hrvoje Benko. Acustico: Surface Tap Detection and Localization using Wrist-based Acoustic TDOA Sensing. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 406–419, Virtual Event USA, October 2020. ACM.

[441] Rui Li, Zhenyu Liu, and Jianrong Tan. A survey on 3d hand pose estimation: Cameras, methods, and datasets. *Pattern Recognition*, 93:251–272, 2019.

[442] Joseph J LaViola Jr. A survey of hand posture and gesture recognition techniques and technology. 1999.

[443] Gabriele Costante, Lorenzo Porzi, Oswald Lanz, Paolo Valigi, and Elisa Ricci. Personalizing a smartwatch-based gesture interface with transfer learning. In *2014 22nd European Signal Processing Conference (EUSIPCO)*, pages 2530–2534. IEEE, 2014.

[444] Hyunchul Lim, Jungmin Chung, Changhoon Oh, SoHyun Park, and Bongwon Suh. Octaring: examining pressure-sensitive multi-touch input on a finger ring device. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 223–224, 2016.

[445] Cheng Zhang, Anandghan Waghmare, Pranav Kundra, Yiming Pu, Scott Gilliland, Thomas Ploetz, Thad E Starner, Omer T Inan, and Gregory D Abowd. Fingersound: Recognizing unistroke thumb gestures using a ring. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):1–19, 2017.

[446] Suranga Nanayakkara, Roy Shilkrot, Kian Peen Yeo, and Pattie Maes. Eyering: a finger-worn input device for seamless interactions with our surroundings. In *Proceedings of the 4th Augmented Human International Conference*, pages 13–20, 2013.

[447] Roger Boldu, Alexandru Dancu, Denys JC Matthies, Thisum Buddhika, Shamane Siriwardhana, and Suranga Nanayakkara. Fingerreader2. 0: Designing and evaluating a wearable finger-worn camera to assist people with visual impairments while shopping. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3):1–19, 2018.

[448] Ke-Yu Chen, Kent Lyons, Sean White, and Shwetak Patel. utrack: 3d input using two magnetic sensors. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 237–244, 2013.

[449] Royu Want, Trevor Pering, Gunner Danneels, Muthu Kumar, Murali Sundar, and John Light. The personal server: Changing the way we think about ubiquitous computing. In *International Conference on Ubiquitous Computing*, pages 194–209. Springer, 2002.

[450] Daniel Ashbrook, Patrick Baudisch, and Sean White. Nenya: subtle and eyes-free mobile input with a magnetically-tracked finger ring. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2043–2046, 2011.

[451] Masa Ogata, Yuta Sugiura, Hirotaka Osawa, and Michita Imai. iring: intelligent ring using infrared reflection. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 131–136, 2012.

[452] Mathias Wilhelm, Daniel Krakowczyk, Frank Trollmann, and Sahin Albayrak. ering: multiple finger gesture recognition with one ring using an electric field. In *Proceedings of the 2nd international Workshop on Sensor-based Activity Recognition and Interaction*, pages 1–6, 2015.

[453] Cheng Zhang, Qiuyue Xue, Anandghan Waghmare, Ruichen Meng, Sumeet Jain, Yizeng Han, Xinyu Li, Kenneth Cunefare, Thomas Ploetz, Thad Starner, Omer Inan, and Gregory D. Abowd. FingerPing: Recognizing Fine-grained Hand Poses using Active Acoustic On-body Sensing. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–10, Montreal QC Canada, April 2018. ACM.

[454] Tengxiang Zhang, Xin Zeng, Yinshuai Zhang, Ke Sun, Yuntao Wang, and Yiqiang Chen. ThermalRing: Gesture and Tag Inputs Enabled by a Thermal Imaging Smart Ring. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, Honolulu HI USA, April 2020. ACM.

[455] Bogdan-Florin Gheran, Jean Vanderdonckt, and Radu-Daniel Vatavu. Gestures for Smart Rings: Empirical Results, Insights, and Design Implications. In *Proceedings of the 2018 Designing Interactive Systems Conference*, pages 623–635, Hong Kong China, June 2018. ACM.

[456] Liwei Chan, Yi-Ling Chen, Chi-Hao Hsieh, Rong-Hao Liang, and Bing-Yu Chen. CyclopsRing: Enabling Whole-Hand and Context-Aware Interactions Through a Fisheye Ring. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 549–556, Charlotte NC USA, November 2015. ACM.

[457] Magic Ring: A Finger-Worn Device for Multiple Appliances Control Using Static Finger Gestures - ProQuest.

[458] Takayuki Iwamoto and Hiroyuki Shinoda. Finger Ring Device for Tactile Sensing and Human Machine Interface. In *SICE Annual Conference 2007*, pages 2132–2136, September 2007.

[459] Boning Zhang, Yiqiang Chen, Yueliang Qian, and Xiangdong Wang. A ring-shaped interactive device for large remote display and mobile device control. In *Proceedings of the 13th international conference on Ubiquitous computing - UbiComp '11*, page 473, Beijing, China, 2011. ACM Press.

[460] Damien Masson, Alix Goguey, Sylvain Malacria, and Géry Casiez. WhichFingers: Identifying Fingers on Touch Surfaces and Keyboards using Vibration Sensors. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pages 41–48, Québec City QC Canada, October 2017. ACM.

[461] Ju Young Oh, Jun Lee, Joong Ho Lee, and Ji Hyung Park. AnywhereTouch: Finger Tracking Method on Arbitrary Surface Using Nailed-Mounted IMU for Mobile HMD. In Constantine Stephanidis, editor, *HCI International 2017 – Posters' Extended Abstracts*, Communications in Computer and Information Science, pages 185–191, Cham, 2017. Springer International Publishing.

[462] Teng Han, Qian Han, Michelle Annett, Fraser Anderson, Da-Yuan Huang, and Xing-Dong Yang. Frictio: Passive Kinesthetic Force Feedback for Smart Ring Output. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, pages 131–142, Québec City QC Canada, October 2017. ACM.

[463] Hsin-Ruey Tsai, Cheng-Yuan Wu, Lee-Ting Huang, and Yi-Ping Hung. ThumbRing: private interactions using one-handed thumb motion input on finger segments. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, pages 791–798, Florence Italy, September 2016. ACM.

[464] Roy Shilkrot, Jochen Huber, Jürgen Steimle, Suranga Nanayakkara, and Pattie Maes. Digital Digits: A Comprehensive Survey of Finger Augmentation Devices. *ACM Computing Surveys*, 48(2):1–29, November 2015.

[465] Hamed Ketabdar, Peyman Moghadam, and Mehran Roshandel. Pingu: A New Miniature Wearable Device for Ubiquitous Computing Environments. In *2012 Sixth International Conference on Complex, Intelligent, and Software Intensive Systems*, pages 502–506, July 2012.

[466] Hamed Ketabdar, Mehran Roshandel, and Kamer Ali Yüksel. MagiWrite: towards touchless digit entry using 3D space around mobile devices. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services - MobileHCI '10*, page 443, Lisbon, Portugal, 2010. ACM Press.

[467] Hamed Ketabdar, Kamer Ali Yüksel, and Mehran Roshandel. MagiTact: interaction with mobile devices based on compass (magnetic) sensor. In *Proceedings of the 15th international conference on Intelligent user interfaces - IUI '10*, page 413, Hong Kong, China, 2010. ACM Press.

[468] Sang Ho Yoon, Yunbo Zhang, Ke Huo, and Karthik Ramani. TRing: Instant and Customizable Interactions with Objects Using an Embedded Magnet and a Finger-Worn Device. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pages 169–181, Tokyo Japan, October 2016. ACM.

[469] Rakhi Sinha, Dolly Uppal, Dharmendra Singh, and Rakesh Rathi. Clickjacking: Existing defenses and some novel approaches. In *2014 International Conference on Signal Propagation and Computer Technology (ICSPCT 2014)*, pages 396–401. IEEE, 2014.

[470] Marc Baloup, Thomas Pietrzak, and Géry Casiez. Raycursor: A 3d pointing facilitation technique based on raycasting. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.

[471] Anil Ufuk Batmaz, Mohammad Rajabi Seraji, Johanna Kneifel, and Wolfgang Stuerzlinger. No jitter please: Effects of rotational and positional jitter on 3d mid-air interaction. In *Proceedings of the Future Technologies Conference (FTC) 2020, Volume 2*, pages 792–808. Springer, 2020.

[472] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 1€ filter: a simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2527–2530, 2012.

[473] Zihao Su, Faysal Hossain Shezan, Yuan Tian, David Evans, and Seongkook Heo. Perception hacking for 2d cursorjacking in virtual reality. 2022.

[474] Benno Roozendaal, Bruce S McEwen, and Sumantra Chattarji. Stress, memory and the amygdala. *Nature Reviews Neuroscience*, 10(6):423–433, 2009.

[475] Sara J Scherr and Jeffrey A McNeely. Biodiversity conservation and agricultural sustainability: towards a new paradigm of 'ecoagriculture'landscapes. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 363(1491):477–494, 2008.

[476] Bernhard Spanlang, Jean-Marie Normand, David Borland, Konstantina Kilteni, Elias Giannopoulos, Ausiàs Pomés, Mar González-Franco, Daniel Perez-Marcos, Jorge Arroyo-Palacios, Xavi Navarro Muncunill, et al. How to build an embodiment lab: achieving body representation illusions in virtual reality. *Frontiers in Robotics and AI*, 1:9, 2014.

[477] Roberto A Montano Murillo, Sriram Subramanian, and Diego Martinez Plasencia. Erg-o: Ergonomic optimization of immersive virtual environments. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*, pages 759–771, 2017.

[478] Joon Gi Shin, Doheon Kim, Chaehan So, and Daniel Saakes. Body follows eye: unobtrusive posture manipulation through a dynamic content position in virtual reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2020.

[479] Michael Rietzler, Florian Geiselhart, Jan Gugenheimer, and Enrico Rukzio. Breaking the tracking: Enabling weight perception using perceivable tracking offsets. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018.

[480] Ying Liu, Stephen J Redmond, Ning Wang, Fernando Blumenkron, Michael R Narayanan, and Nigel H Lovell. Spectral analysis of accelerometry signals from a directed-routine for falls-risk estimation. *IEEE Transactions on Biomedical Engineering*, 58(8):2308–2315, 2011.

[481] Maria Gutierrez-Herrera, Styrmir Saevarsson, Thomas Huber, Joachim Hermsdörfer, and Waltraud Stadler. Repetitive tms in right sensorimotor areas affects the selection and completion of contralateral movements. *Cortex*, 90:46–57, 2017.

[482] Paul Lubos, Gerd Bruder, and Frank Steinicke. Analysis of direct selection in head-mounted display environments. In *2014 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 11–18. IEEE, 2014.

[483] Flavio TP Oliveira, Digby Elliott, and David Goodman. Energy-minimization bias: compensating for intrinsic influence of energy-minimization mechanisms. *Motor control*, 9(1):101–114, 2005.

[484] Xiang Xiao, Huijing Hu, Lifang Li, and Le Li. Comparison of dominant hand to non-dominant hand in conduction of reaching task from 3d kinematic data: Trade-off between successful rate and movement efficiency. *Mathematical Biosciences and Engineering*, 16(3):1611–1624, 2019.

[485] Majid Hajihosseinali, Saeed Behzadipour, Ghorban Taghizadeh, and Farzam Farahmand. Direction-dependency of the kinematic indices in upper extremities motor assessment of stroke patients. *Medical Engineering & Physics*, 108:103880, 2022.

[486] Carolina Cruz-Neira, Daniel J Sandin, and Thomas A DeFanti. Surround-screen projection-based virtual reality: the design and implementation of the cave. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 135–142, 1993.

[487] Lionel Dominjon, Anatole Lécuyer, J-M Burkhardt, Paul Richard, and Simon Richir. Influence of control/display ratio on the perception of mass of manipulated objects in virtual environments. In *IEEE Proceedings. VR 2005. Virtual Reality, 2005.*, pages 19–25. IEEE, 2005.

[488] Scott Frees and G Drew Kessler. Precise and rapid interaction through scaled manipulation in immersive virtual environments. In *IEEE Proceedings. VR 2005. Virtual Reality, 2005.*, pages 99–106. IEEE, 2005.

[489] Victoria Interrante, Brian Ries, and Lee Anderson. Seven league boots: A new metaphor for augmented locomotion through moderately large scale immersive virtual environments. In *2007 IEEE Symposium on 3D User interfaces*. IEEE, 2007.

[490] Barbara La Scaleia, Francesca Ceccarelli, Francesco Lacquaniti, and Myrka Zago. Visuomotor interactions and perceptual judgments in virtual reality simulating different levels of gravity. *Frontiers in Bioengineering and Biotechnology*, 8:76, 2020.

[491] Gad Drori, Paz Bar-Tal, Yonatan Stern, Yair Zvilichovsky, and Roy Salomon. Unreal? investigating the sense of reality and psychotic symptoms with virtual reality. *Journal of Clinical Medicine*, 9(6):1627, 2020.

[492] Eric Burns, Sharif Razzaque, Abigail T Panter, Mary C Whitton, Matthew R McCallus, and Frederick P Brooks. The hand is slower than the eye: A quantitative exploration of visual dominance over proprioception. In *IEEE Proceedings. VR 2005. Virtual Reality, 2005.*, pages 3–10. IEEE, 2005.

[493] Mar Gonzalez-Franco and Jaron Lanier. Model of illusions and virtual reality. *Frontiers in psychology*, 8:1125, 2017.

[494] Jialei Li, Isaac Cho, and Zachary Wartell. Evaluation of cursor offset on 3d selection in vr. In *Proceedings of the 2018 ACM Symposium on Spatial User Interaction*, pages 120–129, 2018.

[495] Brandon J Matthews, Bruce H Thomas, Stewart Von Itzstein, and Ross T Smith. Remapped physical-virtual interfaces with bimanual haptic retargeting. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 19–27. IEEE, 2019.

[496] Shyam Prathish Sargunam, Kasra Rahimi Moghadam, Mohamed Suhail, and Eric D Ragan. Guided head rotation and amplified head rotation: Evaluating semi-natural travel and viewing techniques in virtual reality. In *2017 IEEE Virtual Reality (VR)*, pages 19–28. IEEE, 2017.

[497] Travis Stebbins and Eric D Ragan. Redirecting view rotation in immersive movies with washout filters. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 377–385. IEEE, 2019.

[498] Luv Kohli. Exploiting perceptual illusions to enhance passive haptics. In *IEEE VR Workshop on Perceptual Illusions in Virtual Environments*, pages 22–24. Citeseer, 2009.

[499] Michelle Morón Araújo. Virtual simulators: A tool for current dental education. integrative review. *Universitas Odontologica*, 39, 2020.

[500] Dorota Kamińska, Tomasz Sapiński, Sławomir Wiak, Toomas Tikk, Rain Eric Haamer, Egils Avots, Ahmed Helmi, Cagri Ozcinar, and Gholamreza Anbarjafari. Virtual reality and its applications in education: Survey. *Information*, 10(10):318, 2019.

[501] Narges Ashtari, Andrea Bunt, Joanna McGrenere, Michael Nebeling, and Parmit K Chilana. Creating augmented and virtual reality applications: Current practices, challenges, and opportunities. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pages 1–13, 2020.

[502] Pavel Smutny, Marek Babiuch, and Petr Foltynek. A review of the virtual reality applications in education and training. In *2019 20th International Carpathian Control Conference (ICCC)*, pages 1–4. IEEE, 2019.

[503] Isabell Wohlgenannt, Alexander Simons, and Stefan Stieglitz. Virtual reality. *Business & Information Systems Engineering*, 62:455–461, 2020.

[504] Patrice L Weiss and Adam S Jessel. Virtual reality applications to work. *Work*, 11(3):277–293, 1998.

[505] Elena Márquez Segura, Annika Waern, Jin Moen, and Carolina Johansson. The design space of body games: technological, physical, and social design. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 3365–3374, 2013.

[506] Yupeng Zhang, Teng Han, Zhimin Ren, Nobuyuki Umetani, Xin Tong, Yang Liu, Takaaki Shiratori, and Xiang Cao. Bodyavatar: creating freeform 3d avatars using first-person body gestures. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 387–396, 2013.

[507] Christian Holz and Andrew Wilson. Data miming: inferring spatial object descriptions from human gesture. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 811–820, 2011.

[508] Fraser Anderson, Tovi Grossman, Justin Matejka, and George Fitzmaurice. Youmove: enhancing movement training with an augmented reality mirror. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 311–320, 2013.

[509] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. The go-go interaction technique: non-linear mapping for direct manipulation in vr. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 79–80, 1996.