MULTI-WEAR: A Multi-Wearable Platform For Enhancing Mobile Experiences

Andreas Pamboris[1], Panayiotis Andreou[2], Herodotos Herodotou[3], George Samaras[4]
[1]University of Central Lancashire, apamboris@uclan.ac.uk
[2]University of Central Lancashire, pgandreou@uclan.ac.uk
[3]Cyprus University of Technology, herodotos.herodotou@cut.ac.cy
[4]University of Cyprus, cssamara@cs.ucy.ac.cy

***Abstract*—The uptake of wearable technology suggests that the time is ripe to explore new opportunities for improving mobile experiences. Apps, however, are not keeping up with the pace of technological advancement because wearables are treated as standalone devices, although their individual capabilities better classify them as peripherals with complementary roles. We foresee that the next generation of apps will orchestrate multiple wearable devices to enhance mobile user experiences. However, currently there is limited support for combining heterogeneous devices. This paper introduces MULTI-WEAR, a platform to scaffold the development of apps that span multiple wearables. It demonstrates experimentally how MULTI-WEAR can help bring changes to mobile apps that go beyond conventional practices.**

I. INTRODUCTION

The next big revolution in consumer technology is at our doorstep and comes in the form of wearable devices equipped with computational resources and embedded sensors [1]. These include: (i) *hand wearables* (smart watches and wristbands); (ii) *torso and leg wearables* (smart clothes); (iii) *chest wearables* (typically used for health monitoring); and (iv) *head-mounted wearables* (augmented reality (AR) and virtual reality (VR) glasses/headsets) [2]. Nevertheless, the apps currently developed for such devices are not at the forefront of innovation, *but instead are straitjacketed by limitations of individual platforms, which are targeted separately*.

Different types of wearables usually favor some aspects of mobile user experience (mUX) over others. For instance, hand wearables include a variety of sensors (e.g., accelerometer, ambient light, gyroscope, pedometer, heart rate, and magnetometer), which expose a continuous stream of data related to the user's activity at runtime and/or the execution environment. However, due to form factor constraints, they have significantly smaller displays. Similarly, smart clothes and health wearables support sensing at an even finer granularity by offering head-to-toe coverage of the human body, but they include less means of displaying content. Finally, what AR and VR wearables lack in sensors and flexible input interfaces, they make up for in high-quality 3D visualization capabilities.

Today's wearable apps are constrained by the fact that wearables are treated as standalone devices; developers typically shape their apps around both the positive and negative traits of such a device [3], [4], [5]. Instead, combining different types of wearables together holds the promise of improving mUX. Previous work attempted to combine different wearables to, for e.g., offer accurate location awareness services in AR settings [6] or use voice and gesture recognition simultaneously to achieve a better combined recognition rate [7]. However, building distributed apps on top of heterogeneous platforms entails challenges across the entire development stack. For e.g., developers need to explicitly code the communication between devices and understand how different low-level APIs can be properly coordinated to exploit the combined capabilities of wearables.

This paper presents the design of MULTI-WEAR, a platform to pave the way for streamlining and simplifying the development of *multi-wearable apps*, i.e., apps that leverage multiple wearables, to offer better mUX. MULTI-WEAR aims at abstracting away the complexity involved with orchestrating multiple devices in a single app. It supports a *homogeneous communication layer* to consolidate multiple smart wearables into a single virtual development platform. Furthermore, it provides advanced sensing policies and mechanisms to allow for *querying sensor data* from multiple heterogeneous devices in a unified and efficient manner. Finally, MULTI-WEAR provides *APIs for advanced app capabilities* that leverage the aforementioned layers to make the most of the combined capabilities of different devices (e.g., novel input/output interfaces based on accurate gesture recognition and 3D visual experiences). In summary, the contributions of this paper are:

1) It describes the **architectural design** of the MULTI-WEAR platform, which addresses all major challenges involved with building multi-wearable apps.
2) It **demonstrates experimentally** how MULTI-WEAR can help improve mUX substantially through the development of an e-commerce app on top of a VR headset and a smart wristband; this app is used to evaluate the tangible impact of MULTI-WEAR on user experience.

II. THE NEED FOR WEARABLE INTEGRATION SUPPORT

Table I shows the diversity of current wearables in terms of computational resources, sensors, and programming interfaces, but also provides insights on how they could complement each other when combined through MULTI-WEAR. Hand wearables include a variety of common embedded sensors and typically support input based on voice and touch events. Smart clothing

TABLE I: Comparison of representative devices from different wearable categories

Category	Name	OS	Input Capabilities			Output Capabilities		
			Sensors	Touch	Mic	Display	Audio	Haptic
Hand Wearables	Apple Watch	watchOS	Accelerometer, ambient light, gyroscope, heart rate, NFC	✓	✓	340x272, 390x312	✓	✓
	Moto 360 Sport	Android Wear OS	Accelerometer, ambient light, gyroscope, pedometer, GPS, heart rate, barometer	✓	✓	320x290	✓	✓
	Microsoft Band 2	Microsoft Band 2 OS	Accelerometer, ambient light, gyroscope, barometer, GPS, heart rate, galvanic skin response, skin temperature, UV	✓	✓	320x128	✗	✓
Torso & Leg Wearables	Athos Shirt or Shorts	-	Realtime EMG, heart rate	✗	✗	✗	✗	✗
	Sensoria Socks	-	Pressure sensors	✗	✗	✗	✗	✗
Chest Wearables	ADAMM Asthma monitoring	-	Heart rate, temperature, cough rate, respiration	✗	✗	✗	✗	✓
	QardioCore	-	EKG/ECG, heart rate, temperature, respiratory rate, galvanic skin response	✗	✗	✗	✗	✓
Head Mounted Wearables	Oculus Rift	Oculus Home	Accelerometer, gyroscope, magnetometer	✗	✓	2160x1200	✓	✗
	Vuzix M100	Android ICS 4	Ambient light, GPS, proximity, gyroscope, accelerometer, compass	✗	✓	400x240	✓	✗

items typically don't support local processing, nor do they possess a display of any sorts. Nonetheless, they offer access to new types of sensors such as ECG, breathing rate, and pressure sensors. Similarly, health wearables offer access to even more sensors that, for e.g., measure cough rate, respiration rate, and blood pressure. Finally, head-mounted devices offer less sensors but provide for better means of visualization.

Despite the common characteristics of particular wearable categories, differences within each category also exist, for e.g., in terms of supported connectivity types, display sizes, sensors and computational resources. Combined with the fact that each device runs on top of a different OS and exposes different programming interfaces, this complicates the development of multi-wearable apps. All in all, today's wearable landscape is highly fragmented, which gives rise to significant challenges. Developers are currently required to understand the internals of different wearable platforms and their respective APIs, and to provision explicitly their apps for distributed execution across heterogeneous technologies. This is the main obstacle on the way to realizing future multi-wearable apps.

III. Multi-Ware Architectural Design

The Multi-Wear platform aims at facilitating the development of apps that offer advanced mUX by executing across multiple mobile and wearable devices. Its primary goal is to hide the low-level complexity involved with combining seamlessly different devices. Its architecture (Figure 1) factors functionality of similar responsibilities into separate layers, namely the *Hardware Abstraction*, *Data Management*, and *App Services* layers, which are described next.

A. Hardware Abstraction Layer

The purpose of this layer is to support seamless connection establishment and runtime communication between different types of devices. One device, typically the mobile device, acts as the *mediator*, which is used as a bridge of communication between all other wearables.

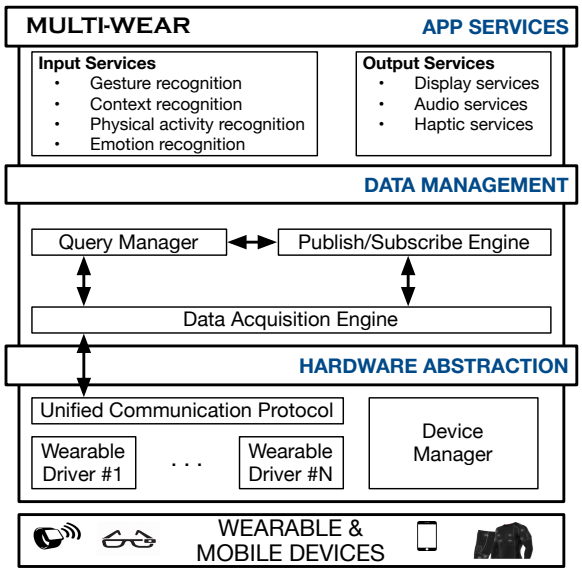


Fig. 1: The Multi-Wear architecture

The *Device Manager* mainly supports connection establishment between different devices at runtime. It supports the discovery of paired devices on the fly by periodically polling for new devices using appropriate discovery libraries (e.g., BandClientManager for Microsoft Band, WCSession for Apple Watch, etc.). It also monitors the status of interconnected devices throughout execution (mainly in terms of availability and power state). This information can be polled by other layers at runtime to adjust accordingly the use of services.

A *Unified Communication Protocol* is used to support the exchange of messages between the mediator and other wearable devices at runtime, over different channels (Bluetooth and WiFi). This protocol allows other Multi-Wear components to contact any connected wearable through a unified, cross-platform interface. Furthermore, it supports three mechanisms

for efficient network communication, to help improve performance in terms of responsiveness and energy consumption. These include: (1) *batch transfers* to allow for reducing the communication overhead associated with multiple short transmissions spaced with small rest periods; (2) *reuse of existing network connections* to avoid connection establishment delays; and (3) *asynchronous communication* to avoid the unnecessary blocking of execution during data transmission.

Finally, the Hardware Abstraction Layer has device-specific functions in place, i.e., the *Wearable Device Drivers*, which allow the mediator to talk to different heterogeneous wearable devices. It maintains a map between the functions provided by the Unified Communication Protocol and the corresponding Wearable Device Drivers, which is used to translate high-level requests (issued by upper layers) into low-level (device-specific) code executed on the supported devices.

B. Data Management Layer

This layer is responsible for managing the flow of data from various embedded wearable sensors to the mediator. It sits in between the Hardware Abstraction and App Services layers as it (1) leverages the former to gain access to heterogeneous physical devices; and (2) services the latter by providing access to data streams in the particular format they are requested.

Query management and data acquisition from multiple wearable sensors are essential operations for building apps with advanced capabilities. Through the *Query Manager*, a developer is provided with the means for (1) creating *queries* to retrieve data from different wearables; and (2) pre-processing (filtering and/or integrating) sensor data streams to conform to the format specified by the requester (an app service).

Two different types of queries are supported: *point* and *continuous* queries. The former consists of one-off requests for a particular sensor reading, while continuous queries are serviced over a longer time span (at specified periodic intervals). To model such queries in a unified way, a SQL-like acquisitional query language is used. For e.g., a query such as Q1: "Retrieve the accelerometer and gyroscope values every 1 second" is represented as follows: `SELECT accelerometer, gyroscope FROM wearables EVERY 1s`. The Query Manager analyzes the syntax of a submitted query, and verifies whether a query can indeed be executed on the available wearables. For e.g., an app service that requires monitoring a user's heart rate could issue Q2: "Retrieve the heart rate of a user 10 times per second". This query however may be constrained by the sampling frequency of the corresponding heart rate sensor. To handle such cases, the Query Manager specifies how to manage infeasible queries (e.g., by raising an appropriate exception to alert apps of MULTI-WEAR's inability to support Q2).

For each query submitted to the Query Manager, a *query object* is initially created, which comprises the following fields: (i) a list of sensors that can provide the data requested (e.g., accelerometer and gyroscope for hand gesture recognition); (ii) the frequency of data retrieval (e.g., every 1s); (iii) the lifetime of a query; and (iv) a condition that triggers the execution of the query (e.g., when heart rate >120 bpm). Query

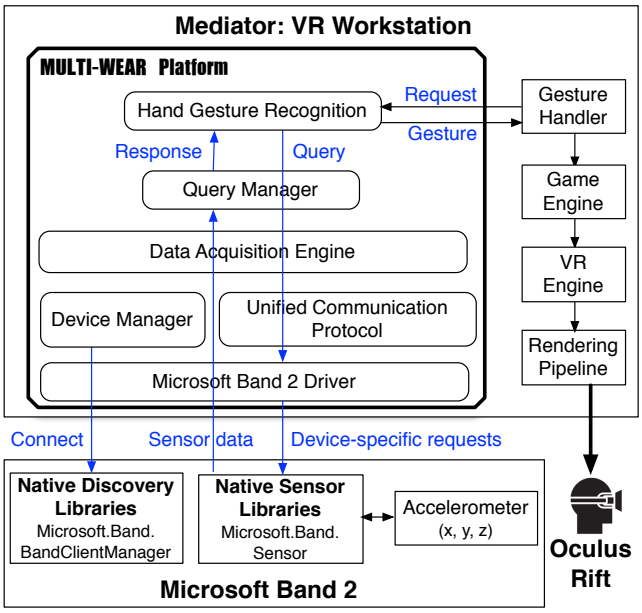


Fig. 2: VRH+W (VR Headset plus Wristband) app structure

objects are later translated into specific requests for different types of supported wearables, using their *Native Sensor Libraries*. This is handled by the corresponding Wearable Device Drivers (in the Hardware Abstraction Layer).

Once queries are handed over to the Hardware Abstraction Layer, the *Data Acquisition Engine* is responsible for registering listeners that handle events raised by different wearables, e.g., when new measurements are made available. In the case of Q1, two such listeners are registered to account for new accelerometer and gyroscope readings, respectively. The measurements collected eventually reach the Query Manager, which is responsible for combining them into a *result object* that is returned to the requesting service.

The Data Management Layer also supports more efficient query execution plans when multiple interleaving (continuous) queries are in progress. This is achieved through the *Publish Subscribe Engine*, which sits in between the Query Manager and the Data Acquisition Engine, and is used for caching and sharing results between queries with similar requirements. The Query Manager can avoid going all the way down to the wearables if the requested data can be deduced from the results of existing active queries. For e.g., two concurrent app services may require access to the same sensor data. Assuming the addition of Q3: "Retrieve the accelerometer and gyroscope values every 5 seconds", the Query Manager is able to detect that Q3 can be answered by sampling the results of Q1; the results of Q1 are published to the Publish/Subscribe Engine, which allows Q3 to reuse these results by subscribing to Q1.

C. App Services Layer

This layer is responsible for providing high-level services that can be used as building blocks to realise novel app functionality. It currently offers two types of services for creating advanced *input* and *output* app components.

Output Services expose interfaces for creating and managing notifications in various formats (e.g., long messages, quick responses, alerts), which span multiple devices. They support various delivery methods such as displaying text or images on multiple device screens as well as haptic and audio feedback.

Input Services can be used for providing input to an app in new novel ways. This component offers an API for accessing useful information regarding the user and execution environment at runtime, which can trigger different events depending on the core business logic of an app. Input services can be split conceptually into two categories: *explicit* and *implicit* services, depending on whether they rely on actions initiated explicitly by the user (instructing the app to perform some task), or input inferred implicitly by MULTI-WEAR.

Explicit input services include: (1) *Gesture Recognition* services, which recognize user gestures (for e.g., hand movements used to trigger a change of the user's field of view in a virtual reality environment); and (2) services that leverage interchangeable input methods based on voice, hardware buttons, touch screen actions, etc. (supported collectively by a wide range of wearables).

Implicit input services include: (1) *Context Recognition* services, which infer knowledge about the ambient execution environment (for e.g., light and sound levels in a room based on ambient light sensors and microphones, or user location based on GPS sensors); (2) *Physical Activity Recognition* services, which detect user physical statuses (for e.g., standing, walking, running, sitting, lying down, etc., based on accelerometer, gyroscope, and heart rate sensors); and (3) *Emotion Recognition* services, which detect user emotional states (for e.g., stress levels, based on heart rate and galvanic skin response sensors). Such input information can be used to allow apps to react to user and/or environment changes on the fly.

IV. THE CASE OF A MULTI-WEARABLE E-COMMERCE APP

This section demonstrates experimentally how MULTI-WEAR can be used to improve an app's capabilities by leveraging multiple wearable devices. In particular, it compares two versions of an e-commerce app, which were developed for: (i) a VR headset only (VRH); and (ii) a VR headset combined with a smart wristband (VRH+W) using MULTI-WEAR.

A. E-commerce Use Case

The e-commerce app implements an online virtual store that supports the following functionality: users can navigate through the store; browse clothing items; select/view a specific item; customize a selected item according to size and color preferences; and add/remove different items to/from their shopping cart. Two modes of execution are supported: (i) the *navigation* mode, which allows users to move within the confinements of the virtual store, and (ii) the *select* mode, which allows users to select clothing items and menu options.

The first version of the app, termed VRH, is implemented on top of an Oculus Rift Development Kit 2 (DK2) headset[1] and a VR workstation. In *navigation* mode, users are able to move within the virtual store using head movements to point at directional arrows overlayed on the screen. To switch to *select* mode, users focus on an item for two seconds using a tracker in the center of the screen. The two-second delay is required to ensure that no accidental mode switching occurs at runtime.

The second version, termed VRH+W, uses additionally a Microsoft Band 2 wristband[2] and a Bluetooth 4.0 LE transceiver, which allows for connecting the wristband with the VR workstation. The wristband allows users to navigate using hand gestures by swiping their hand accordingly in different directions. In addition, hand gestures are used to implement shortcuts for frequent operations such as changing the size and color of a selected item, adding an item to the shopping cart, and showing or hiding the shopping cart.

To develop the VR counterpart of both app versions we used the Unity game engine[3], which supports the Oculus Rift headset. For VRH+W, we also used the Band SDK for the Universal Windows Platform in C# to program the Microsoft Band 2 wristband. As part of the MULTI-WEAR platform, a gesture recognition model for the wristband was implemented based on the open-source Wiigee project[4], which employs a Hidden Markov Model to train and recognize user-specified gestures. Finally, communication between the devices was implemented using a SignalR-based[5] communication platform; the platform supports communication between programs executing in different environments (Java, .NET 2.0, .NetCore).

Figure 2 describes the architecture of the VRH+W app built on top of MULTI-WEAR. It leverages a subset of the full-fledged platform (as described in §III), which includes a prototype implementation of: the *Device Manager* to handle connection establishment between the wearable devices and the VR workstation; the *Query Manager* and the *Data Acquisition Engine* to manage the continuous retrieval of accelerometer data from the wristband; and *App Services* for recognizing hand gestures based on accelerometer sensor data. Provided these components, a developer is asked to implement the remaining app functionality, which relates to the VR and Game engines, as well as the app logic for reacting to different hand gestures that are recognized seamlessly through MULTI-WEAR.

B. Experimental Setup

We compare VRH with VRH+W by conducting experiments with a group of users. The goal of the experiments is to demonstrate how the combination of the VR headset and the wristband through MULTI-WEAR can significantly improve mUX. In particular, for the e-commerce use case, we identify improvements along two main dimensions: *execution time* and the *quality of user interaction* with the virtual environment.

Users: A total of five computer science students participated in the experiments (4 males, 1 female, mean age 21.4). All participants had prior experience in using VR systems.

[1]Oculus Rift, https://www.oculus.com/en-us/

[2]https://www.microsoft.com/microsoft-band/en-gb

[3]Unity Game Engine, http://unity3d.com/

[4]http://www.wiigee.org/

[5]SignalR, http://signalr.net/

TABLE II: App scenarios used in experiments

Scenario	Steps
S.{a,b,c} (Differ in target location)	1. Navigate to a target location in the store 2. Select an item from a stand 3. Select a color 4. Add the item to cart 5. Select a size 6. De-select the item
S.d	1. Open cart 2. View cart 3. Close cart

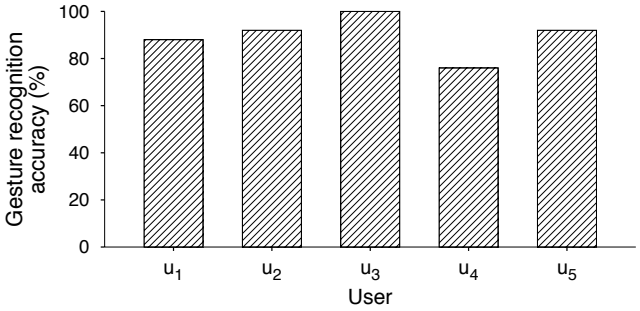
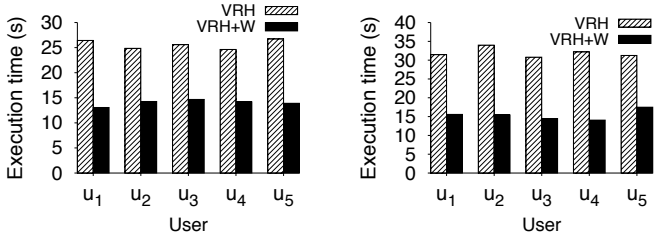
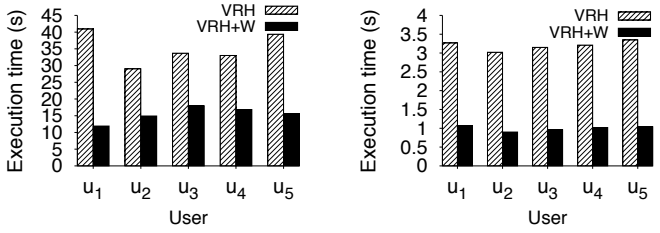


Fig. 4: Accuracy of gesture recognition

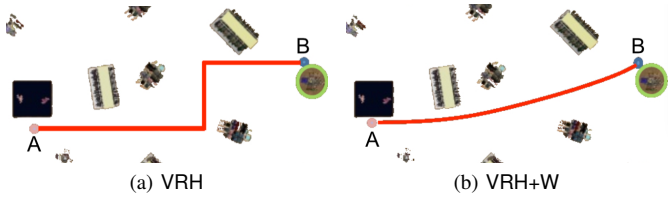


(a) Scenario S.a (b) Scenario S.b



(c) Scenario S.c (d) Scenario S.d

Fig. 3: Scenario completion times using VRH Vs. VRH+W



(a) VRH (b) VRH+W

Fig. 5: User trajectory on VR ground view map for S.a

App scenarios: Table II describes the four scenarios used in experiments, chosen to cover different aspects of the virtual shop experience. These include: locating an item, changing its size and color, adding items to and viewing the shopping cart. Scenarios S.a–c cover different movement patterns and differ only in the virtual location users are asked to navigate to.

Experiments: (1) participants are asked to execute all four app scenarios in both VRH and VRH+W to compare execution times and the quality of user experience; and (2) participants are asked to perform 25 hand gestures (e.g., swiping in different directions) to record the accuracy of MULTI-WEAR's gesture recognition. In all experiments, we measure the duration of each step involved and record the ground view map of the virtual environment using an external monitor.

C. Execution Time

Figure 3 shows the execution time per app scenario (described in Table II) for each of the users u_i ($1 \leq i \leq 5$) who participated in the experiments. An average reduction in execution time by approximately 45–56% for scenarios S.a–c, and 69% for scenario S.d was achieved, with negligible variations across different experiments. The S.a–c results show that by using MULTI-WEAR's input services based on hand gestures, navigation is much faster as it avoids the overhead associated with having to point at overlay arrows using head movements in order to change the direction of movement. Furthermore, VRH+W's shortcuts (again based on hand gesture inputs) for changing the color or size of a selected item, and for adding an item to the shopping cart, yield additional savings in execution time. This is because they avoid the artificial delay used in VRH to distinguish between such actions and head movements that aim at changing a user's viewpoint. The impact of shortcuts for showing and hiding the shopping cart is even more compelling, as indicated by the results for S.d.

D. Interaction with Virtual Environment

Next, we evaluate the quality of user interaction with VRH and VRH+W by comparing the navigation patterns of users that move within the confinements of the virtual store. We also evaluate the accuracy of the gesture recognition model used to replace less intuitive navigation controls like hovering over overlay arrows using head movements (used in VRH).

1) Quality of User Navigation: To wholly immerse users in a VR experience, apps need to support both realistic 3D visualization and interaction between users and the virtual environment. Based solely on the Oculus Rift, VR experience covers only the visual side of things—as reported by its inventor, VR apps for Oculus Rift currently offer suboptimal experiences by lacking a fully integrated input/output system that can provide for a natural way to interact with the virtual world [8].

Combining the Oculus Rift with a wristband wearable is a step towards improving significantly the interaction with the virtual environment. This experiment focuses on the quality of user navigation by comparing the movement patterns of users in VRH and VRH+W. Figure 5 shows a ground view map of the virtual store, annotated with the trajectory of a user who is asked to move between two pre-defined locations. In the

case of VRH, users are required to hover over overlay arrows using head movements in order to move in different directions. This, however, prevents them from being able to change their viewpoint simultaneously, i.e., turn their head in different directions to view their virtual surroundings. As a result, users cannot change the direction of movement while being on the move, which is why a user's trajectory comprises straight connected lines. In contrast, VRH+W supports navigation (using hand gestures) without compromising one's ability to use head movements to switch viewpoint simultaneously, which is reflected by the more natural curved path shown in Figure 5b.

2) Gesture Recognition Accuracy: The prototype gesture recognition model used in VRH+W is implemented as an Input service in MULTI-WEAR's App Services layer. To determine the model's accuracy, each user was asked to perform 25 gestures from a vocabulary of six hand gestures, which cover different directions of swiping (up, down, left, right, forward and backwards). The results of this experiment (Table 4) show a success rate that ranges between 76–100%, averaging to approximately 90% across all users.

V. RELATED WORK

Previous work has focused on special-purpose systems developed from the ground up with the intention of leveraging multiple ambient and wearable devices. The work by Kourogi et al. [6] proposes an AR system that combines multiple wearable devices to enable accurate location and orientation awareness services. Such wearables include cameras, attitude sensors, accelerometers, magnetometers, gyro-sensors, and inclinometers, all attached to the user's body. In health care, proposed systems [9], [10], [11] combine ambient and wearable sensors (e.g., ear-worn and blob-based vision sensors, accelerometers, gyroscopes, and bio-signal sensors) to enable the robust recognition of activities and motions. Similarly, the work by Cho et al. [7] proposes a distributed system that comprises a wristband-type device and a mobile gateway. The system supports voice and gesture recognition simultaneously in order to achieve a better combined recognition rate. PalmType [12] combines wrist-worn sensors with smart glasses in order to use palms as keyboards for text entry.

More related to MULTI-WEAR is the work by Brito et al. [13] on a service-oriented sensor middleware architecture for data acquisition and processing in the health domain. This middleware addresses requirements for sensor interoperability in heterogeneous medical devices, dealing specifically with challenges related to data collection and aggregation. Uddin et al. [14] propose a framework for monitoring human activity using wearable devices, which focuses on reducing the data processing and energy overheads of the wearable device by preprocessing and filtering out data collected by the device.

The aforementioned systems are designed explicitly to support specific devices and tasks such as accurate position tracking, better motion detection, or voice/gesture recognition. They embody a narrow set of features focused primarily around measurement refinements for better accuracy, or efficient data collection and aggregation mechanisms. Furthermore, they cannot be easily extended to leverage arbitrary wearable devices, and mainly consider wearable sensors as opposed to full-fledged devices. In contrast, MULTI-WEAR is a general-purpose platform for combining, at different levels of abstraction, arbitrary types of wearables in order to facilitate the development of any kind of multi-wearable apps.

VI. CONCLUSIONS

The future of mobile apps lies with the fusion of multiple wearables to benefit from their combined capabilities. MULTI-WEAR is a platform that provides the necessary abstractions to support developers in the integration of multiple wearables under the umbrella of a single app. To validate the impact of multi-wearable apps on user experience, using a prototype implementation of MULTI-WEAR, we implemented and evaluated two versions of a VR app (one based solely on a VR headset and another that combines a VR headset with a smart wristband). We compared them experimentally to show that the latter can significantly improve mobile user experience.

ACKNOWLEDGMENTS

This work was partially supported by the European H2020 project GrowMeUp (#643647). We would like to thank the InSPIRE center for its contribution to the implementation and evaluation of this work.

REFERENCES

[1] Business Insider, *THE WEARABLES REPORT: Growth Trends, Consumer Attitudes, and Why Smartwatches will Dominate*, 2015, https://goo.gl/HxeU58.

[2] Vandrico Inc., *The Wearable Database*, 2017, https://vandrico.com/wearables/.

[3] Max Castleman, *The Advantages and Disadvantages of Wearable Tech*, 2014, https://goo.gl/nH2cuj.

[4] Insurance Journal, *Pros and Cons of Wearable Technology in the Workplace*, 2015, http://www.insurancejournal.com/news/national/2015/08/07/377825.htm.

[5] InformationWeek, *Smartwatches Still Lack Killer App*, 2016, https://goo.gl/74kie3.

[6] M. Kourogi and T. Kurata, "Personal Positioning Based on Walking Locomotion Analysis with Self-Contained Sensors and a Wearable Camera," in *ISMAR*, 2003.

[7] I.-Y. Cho, J. Sunwoo, H.-T. Jeong, Y.-K. Son, H.-J. Ahn, D.-W. Lee, D.-W. Han, and C.-H. Lee, "A Distributed Wearable System Based on Multimodal Fusion," in *ICESS*, 2007.

[8] The Verge, *Virtual Reality Check: Why Controllers Haven't Caught Up to the Oculus Rift*, 2013, http://goo.gl/sfrAoq.

[9] J. Pansiot, D. Stoyanov, D. McIlwraith, B. Lo, and G.-Z. Yang, "Ambient and Wearable Sensor Fusion for Activity Recognition in Healthcare Monitoring Systems," in *BSN*, 2007.

[10] D. G. McIlwraith, J. Pansiot, and G. Yang, "Wearable and Ambient Sensor Fusion for the Characterisation of Human Motion," in *IROS*, 2010.

[11] J.-K. Min and S.-B. Cho, "Activity Recognition Based on Wearable Sensors Using Selection/Fusion Hybrid Ensemble," in *SMC*, 2011.

[12] C.-Y. Wang, W.-C. Chu, P.-T. Chiu, M.-C. Hsiu, Y.-H. Chiang, and M. Y. Chen, "PalmType: Using Palms As Keyboards for Smart Glasses," in *Proc. of the 17th Intl. Conf. on Human-Computer Interaction with Mobile Devices and Services*, 2015.

[13] M. Brito, L. D. Vale, P. Carvalho, and J. Henriques, "A Sensor Middleware for Integration of Heterogeneous Medical Devices," in *Proc. of the 32nd Annual Intl. Conf. of the IEEE Engineering in Medicine and Biology Society*, 2010.

[14] M. Uddin, A. Salem, I. Nam, and T. Nadeem, "Wearable Sensing Framework for Human Activity Monitoring," in *Proc. of the 2015 Workshop on Wearable Systems and Applications (WearSys)*, 2015.