Signal-processing Transformation from Smartwatch to Arm Movement Gestures

Franca Rupprecht[1], Bo Heck[1], Bernd Hamann[2] and Achim Ebert[1]

[1] Technische Universität Kaiserslautern, Computer Graphics and HCI,
67663 Kaiserslautern, Germany
{rupprecht, ebert}@cs.uni-kl.de
bheck@rhrk.uni-kl.de

[2]University of California, Department of Computer Science,
Davis, CA 95616, USA
hamann@cs.ucdavis.edu

Abstract. This paper concerns virtual reality (VR) environments and innovative, natural interaction techniques for them. The presented research was driven by the goal to enable users to invoke actions with their body physically, causing the correct action of the VR environment. The paper introduces a system that tracks a user's movements that are recognized as specific gestures. Smartwatches are promising new devices enabling new modes of interaction. They can support natural, hands-free interaction. The presented effort is concerned with the replacement of common touch input gestures with body movement gestures. Missing or insufficiently precise sensor data are a challenge, e.g., gyroscope and magnetometer data. This data is needed, together with acceleration data, to compute orientation and motion of the device. A transformation of recorded smartwatch data to arm movement gestures is introduced, involving data smoothing and gesture state machines.

Keywords: Intuitive and natural interaction · Low budget interaction devices · Mobile devices · Virtual Reality · Body movement gestures · Gesture recognition

1 Introduction

Mobile devices are almost ambiguous today and feature a wide range of input and output capabilities like touch screens, cameras, accelerometer, microphones, speakers, near-field communication, Wi-Fi, etc. The usage of smart-devices is easy and intuitive, and they offer a wide range of interaction metaphors, which can lead to a more natural and intuitive interaction as well as a broad array of control elements [1]. Especially the smartwatch as latest technology in that field gives new possibilities of interaction techniques. As the watch is fix on the wrist, the hands are free what leads to a more natural interaction in the meaning of body gestures, also other technology like finger tracking can be combined and new interaction techniques will be enabled. Next to the

common touch gestures which are performed very frequently on the smart device's display we developed additional movement gestures which enriches the input capabilities of the smartwatch significant. Virtual Reality (VR) visual interaction environments make possible the sensation of being physically present in a non-physical world [2]. The value of this experience is a better perception and comprehension of complex data based on simulation and visualization from a near-real-world perspective [3]. A user's sense of immersion, the perception of being physically present in a non-physical world, increases when the used devices are efficient, intuitive, and as ”natural” as possible. The most natural and intuitive way to interact with data in a VR environment is to perform the actual real-world interaction [4]. For example, gamers are typically clicking the same mouse button to swing a sword in different directions. However, the natural interaction to swing a sword in a VR application is to actually swing the arm in the physically correct direction as the sword is an extension of the user's arm. Therefore, intuitive and natural interaction techniques for VR applications can be achieved by using the human body itself as an input device [5]. VR devices are usually specialized to support one interaction modality used only in VR environments. Substantial research has been done in this field, yet VR input devices still lack highly desirable intuitive, natural, and multi-modal interaction capabilities, offered at reasonable, low cost. In a preliminary study [6] we figured out that our low budget setup and the implemented in air gestures are comparable to common VR input technology, specifically body tracking enabled by a 3D camera. Thereby, we could state that a combination of smartphone and smartwatch capabilities, outperforming a comparable common VR input device. We have demonstrated the effective use for a simple application. The main advantages of our framework for highly effective and intuitive gesture-based interaction are:

- Location independence
- Simplicity-of-Use
- Intuitive usability
- Eyes-free interaction capability
- Support for several different inputs
- High degree of flexibility
- Potential to reduce motion sickness
- Elegant combination with existing input technology

In this paper, we present the improvements of our gesture recognition algorithms and the adaption of enhanced gestures accordingly. For the investigation, the Apple Watch (watchOS2) is used to develop different arm movement gestures to enable new natural interaction mechanisms. Unfortunately, the only sensor data which can be proceed from the Apple Watch are the acceleration data as gyroscope and magnetometer was not accessible to the time of our approach. Those three sensors are used to calculate the orientation and motion dynamics of the device. The challenge is described by missing sensor data, precisely gyroscope and magnetometer data, which need to be compensated in order to calculate orientation and motion dynamics of the device. The aim of this paper is to create a system that is able to recognize arm gestures only using the accelerometer data of the device. This system has to allow a person, skilled in programming, to define their own gestures. Based on six key values and a statement sequence we are able to define precise arm movement gestures, exemplary demonstrated on seven different gestures. More gestures are conceivable and easily adoptable with our approach. Subsequent, we are able to transform the giving signal-processing from the smartwatch into arm movement gestures with the use of smoothing

algorithms and gesture state machines which lead to the actual gesture recognition. After giving a short description of the signal-processing model, and the associated existing shortfalls, we will demonstrate the resulting adaption of gestures in detail. Furthermore, an evaluation of the gesture recognizer is conducted and the results are presented.

2 Related Work

Current research covers many aspects of interaction in VRs, being of great interest to our work. Bergé et al. [7] stated that mid-air hand and mid-air phone gestures perform better than touchscreen input implying that users were able to per- form the tasks without training. Tregillus et al. [8] affirmed that walking-in-place as a natural and immersive way to navigate in VR potentially reduce VRISE (Virtual reality induced symptoms and effects [9]) but they also address difficulties that come along with the implementation of this interaction technique. Freeman et al. [10] addressed the issue of missing awareness of the physical space when performing in-air gestures with a multi-modal feedback system. In order to overcome the lack of current display touch sensors to equip a user with further input manipulators, Wilkinson et al. utilized wrist-worn motion sensors as additional input devices [11]. Driven by the limited input space of common smart watches, the de- sign of non-touchscreen gestures are examined [12]. Houben et al. prototyped cross-device applications with a focus on smartwatches. In their work, they provided a toolkit to accelerate application development process by using hardware emulations and a UI framework [13]. Similar to this work, several investigations concerning interaction techniques with wrist-worn devices such as smartwatches and fitness trackers have been made. In general, two types of recognizing techniques can be differentiated: 1) machine learning techniques base on a (high) number of training samples from which features are extracted and gestures with the use of probability classifiers identified and 2) simple pattern recognition with predefined features. Mace et al. [14] compared naive Bayesian classification with feature separability weighting against dynamic time warping. The extremely differing gesture types (circle, figure eight, square, and star) could be recognized with an average accuracy of 97% for the feature separability weighted Bayesian Classifier, and 95% for the dynamic time warping with only five gesture samples. Mänyjärvi et al. [15] presented a hidden markov model in order to define continuous gesture recognition based on user defined gestures for primitive gestures used to remotely control a DVD player. Their model could reach an accuracy value of 90% to 95%. The investigation performed in [16] employs a hidden markov model for user dependent gesture recognition. The models are used for training and recognition of user-chosen gestures performed with a Wii remote controller. Only few gestures are tested, which differ extremely. Shortcomings of high computational power are mentioned. Compared to the Wii remote controller, the smartwatch data does not show comparable high peaks. Thus, the model is not suitable for the underlying kind of data. Methods from machine learning have a high flexibility and find application especially at end user side, where user do not know the data nor are able to identify features. Those techniques can lead to excellent accuracy rates with an exceeding number of training samples. However, classifying the training data and identifying the correct gesture with machine learning

techniques are resource-intensive. Considering, the less computation power of a smartwatch and that the end user is not the point of interest in this work, machine learning techniques are not applicable. Work done in [19] presents a frame-based feature extraction stage to accelerometer-based gesture recognition performed with the Wii remote controller. A gesture descriptor combining spectral features and temporal features is presented. However, the recognition starting point is activated with a mouse click and not only due to the recognition. Chu et al. [17] used a fuzzy control technique to classify different gestures based on acceleration data from smartphones. The gestures defined in the study are totally different, it would be interesting if there is still a precision rate of 91% with gestures more similar to each other.

3 Setup

Our approach uses common technologies, at relatively low cost, supporting intuitive, basic interaction techniques already known. A smartphone fixed in an HD viewer serves as fully operational HMD and allows one to experience a virtual environment in 3D space. The smartphone holds the VR application and communicates directly with a smartwatch. Wearing a smartwatch with in-built sensors "moves" the user into the interaction device and leads to a more natural interaction experience. In order to support control capabilities to a great extent, we consider all input capabilities supported by the smartphone and the smartwatch. In addition to touch display and crown, we considered accelerometer, gyroscope and magnetometer, as they are built-in sensors. Our watch setup consists of two components: (1) A smart- watch, the Apple Watch Sport 38mm Generation 1 and (2) an HMD. The watch's dimensions are 38.6mm x 33.3mm x 10.5mm. Neither watch nor HMD are tethered, and there is no technical limitation to the tracking area. Also, the battery is no limiting factor in our investigation. A user's range of movement is defined by the actual physical space. One considerable limitation is the fact that body movement gestures are limited to one arm. This limitation implies that all other body parts cannot be utilized for gesturing. Body movements and gestures involving more body parts, like legs, both arms, or torso, would enable a more natural user interface experience. Smart watch and smartphone are connected in our framework via Bluetooth, making possible a continuous communication. Accelerometer data collected by the watch are communicated to the phone that computes and detects defined gestures, making use of the smartphone's computation power. It is challenging to devise an algorithm to transform the raw stream of accelerometer data into explicit gestures. Gestures should not interfere with each other, and the system must compute and detect gestures in real time. The resulting data stream to be transmitted and the resulting computation time required for data processing can lead to potential bottlenecks. In previous work, we designed seven distinct gestures dedicated to VR modes of orientation, navigation, and manipulation. evaluated them comparatively to common VR input technology, specifically body tracking enabled by a 3D camera. During the experiment the user was located in a VE constituting a factory building. Latter is an accurate 3D model of a machine hall existing in real world. **Orientation** is implemented through head-tracking. A user can look around and orientate oneself. The smartphone uses built-in sensors, like accelerometer and gyroscope, to deter- mine orientation and motion (of the devices), permitting translation, done by the game

engine, into the user's viewpoint in a virtual scene. **Navigation** is implemented by two interaction techniques: (1) In the watch setup, a user looks in walking direction, and single-touch taps the watch to indicate begin or end of movement. (2) In the 3D Camera setup, a user "walks on the spot". **Manipulation** refers to the interaction with objects in a scene.

4 System Model

In order to describe the model, we define a gesture as following: A gesture is a pattern of wrist movements. Patterns of interest are characterized as intentionally performed, easily memorable, and easily performed by a wide range of users. Furthermore, a gesture pattern is a sequence of states based on key information of the sensor data. Our approach based on one central class *GestureRecognizer*, that collects, refines and translates the sensor data. Every gesture is an object of the type *Gesture*, that implements a state machine (SM), that takes the refined sensor data and per- forms state changes accordingly. To enable the recognition of a specific gesture, the corresponding class gets registered in the `GestureRecognizer`, so that its update gets called periodically. The recognized gesture is send to the `UnityEngine`, where corresponding functionality is executed. In terms of computation time and performance, it is recommended to register only the gestures that should be performed and as long as needed. Afterwards they should be released again.

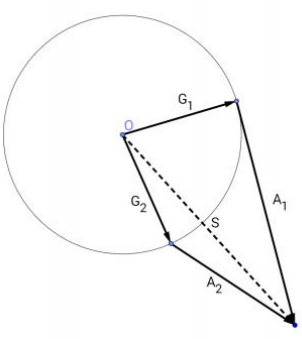
State Machine. A state machine (SM) is a model that describes the output of a control system based on the incoming stimuli from the past. States represent all possible situations in which the state machine may ever be. The incoming inputs and the sequence of inputs in the past determine the state in the system and lead to the corresponding output of reaching that state [18]. If the number of distinguishable situations for a given state machine is finite, the number of states is finite and the model is called a finite state machine.

Key values. Every time a state machine gets updated, the following key values are used to determine the state and corresponding transition:

- Direction of Acceleration
- Direction of Velocity
- Direction of Gravity
- Value of Acceleration
- Value of Velocity
- Time

In each frame the extracted sensor data are sent to all state machines (SMs) as new incoming inputs. In general, an update of the SMs occurs every time new input arrives; some SMs only get updated if a key value changes. An update of the SM does not imply a state change. Two different types of SMs are used in the system model. The first type of SM defines states based on segment positions. The second type of SM defines states based on the number of reached segments. For the purpose of the definition, they are both true SMs, but we can use a lot less states this way, as some gestures do not need an exact position but a specific number of direction changes. If one gesture is recognized, all SMs get reset and a message is sent to execute the desired interaction in the VR environment. As condition of our approach, we merely use the patterns

generated by a single axis accelerometer and try to extract the information needed to define gestures. The key values are calculated based on the following sensor data: (1) gravitation, (2) acceleration, and (3) velocity. The gravitation value is extracted in order to be used as reference of the watch's posture with which we can align the sensor data and ultimately because the gravitation is polluting the sensor-data. Acceleration data is used to compute the path of the wrist in 3D space monitored over time. The velocity is derived from the acceleration data and used to define additional state transition conditions. An overview of all forces and their dependencies are depicted in Fig. 1.



$\vec{S} = Raw\ sensor\ data$
$\vec{G} = Gravitation$
$\vec{A} = Acceleration$
$\vec{V} = Velocity$

We know:

$\vec{S} = \vec{G} + \vec{A}$
$\|G\| = 1$

Fig. 1. Key values of gesture recognizer. Different $\vec{G}$ and $\vec{A}$ can add up to $\vec{S}$.

If we measure $\vec{S}$ then we know that $\vec{G}$ also points in the same direction, meaning $\vec{G} = \frac{\vec{S}}{\|S\|}$. However, adopting the direction of $\vec{G}$ from $\vec{S}$ is only correct if the watch stands still. Therefore, it is challenging to find the right moment to calculate the direction. Our approach is to implement an adoption-rate describing the degree of how much we trust in $\frac{\vec{S}}{\|\vec{S}\|}$ to be the same as the actual/real gravitation force with the following steps:

$$\vec{G}_{new} = \vec{G}_{old}\,(1 - weight) + weight\ \frac{\vec{S}}{\|\vec{S}\|} \tag{1}$$

$$weight = 0.3\ \cdot \exp\left(-\left(\|\vec{S}\| - 1\right)^2 \cdot 14\right) \tag{2}$$

The *weight* distribution corresponds to a Gaussian bell curve, which has its peak in $\|\vec{S}\| = 1$. This function guarantees that $\vec{G}$ is rapidly corrected once the user stays still and that the changing rate from $\vec{G}$ is lowered while gestures are performed. Computations according to Equation 1 are performed every frame with a rate of frames per second; after merely frames G_{old} is only covered by $(1 - 0.3)^8 = 5{,}7\%$ if $\|\vec{S}\|$ is close to 1. After computing $\vec{G}$ we know:

$$\vec{A} = \vec{S} - \vec{G}_{new} \tag{3}$$

This approximation still does not consider the position of the smartwatch on the wrist and therefore $\vec{G}_{new}$ could be pointing anywhere. Taking this into account, we apply a

rotation R to $\vec{A}$ with the property (0, 0, -1) = $\frac{\vec{G}_{new}}{\|\vec{G}_{new}\|} \cdot R$. Thus, the robustness of the gesture recognition is enhanced.

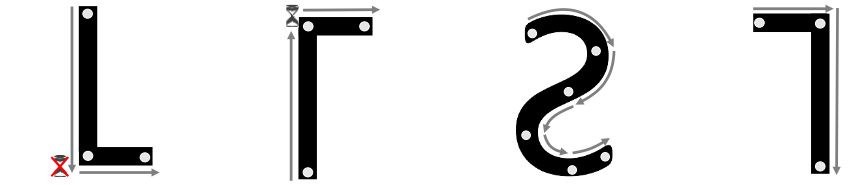
States. In order to define states for the state machine in an easy computable form, areas on a sphere are defined into sectors. Every vector is transformed into an identifier representing if this vector lies in that sector. In total, we defined 28 sectors: 5 sectors in longitude axis, whereby the 3 middle slices are divided into 8 sectors in latitude axis. Tested in a preliminary study, we figured out that those sectors are the optimal number of segmentations, which are comfortable to reach and that provide an adequate number of possible permutations and therefore gesture states. Based on the defined sectors, the following refined denotations of the key values are used: $Acceleration =$ Sector to which $\vec{A}$ points to; $Velocity =$ Sector to which $\vec{V}$ points to; $Gravity =$ Sector to which $\vec{G}$ points to; $\|Acceleration\| =$ Value of Acceleration $= \|\vec{A}\|$; $\|Velocity\| =$ Value of Velocity $= \|\vec{V}\|$. Next, due to the keen accelerometer sensors and repetitive assimilation of data in- accuracies over time, so-called drift of the computed acceleration occurs. This well-known problem appears by using sensors without the ability to re-calibrate. Multiple factors lead to this inaccuracy. The system is often slightly lagging behind a move- ment and hence computes faulty values. If the hand is rotated by degree, the sign of the number changes. As the systems sensor is slightly lagging behind, the sign shift is not recognized for a short time. But we cannot avoid the sensor lag, as we cannot trust that the direction of $\vec{G}$ and the direction of $\vec{S}$ is the same. As defined, $\vec{S} = \vec{A}_{real} + \vec{G}$ and $\vec{A}_{polluted} = \vec{S} - \vec{G}$, for that short moment, one has $\vec{A}_{polluted} = \vec{S} + \vec{G} = \vec{A}_{real} + 2\vec{G}$, and $\vec{G}$gets adjusted. Furthermore, over time the $\sum \vec{A} \neq 0$, therefore the velocity of the object, also is $\|Vel\| \neq 0$. In long movements, the velocity drifts extremely as the sensor cannot be calibrated $\vec{G}$ is not adjusted. It cannot be stressed enough that rotating the wrist causes anomalies due to above mentioned problem, which implies that $\vec{A}$ and $\vec{V}$ cannot be trusted for / of a second after a full rotation is performed. In order to effect higher accuracy, the following adjustment to the velocity vector is performed every frame:

$$\vec{V}_{new} = \vec{V}_{old} \cdot reductionFactor_1 + \vec{A} \cdot timeFactor - reductionFactor_2 \qquad (4)$$

5 Gesture Design

We defined seven gestures that cover the full range of possible gestures to evaluate the gesture recognition algorithm. Hereby, it is implied, that single states are recognized as well as sequences of states and changes of the key values. These gestures also try to prove that a series of movements can make up a recognizable pattern. We show this for a realistic number of steps. The classification of the gestures follows along the attributes of movement and shape. We differentiate motion between continuous and partitioned movements. If a gesture is performed without breaks it has a continuous movement while partitioned gestures are made up of a series of continuous sub-gestures with sufficient breaks in-between. The second differentiation between gestures is related to

the gesture form. Gestures have either curvy or angular paths describing their shape, see Fig. 2. Following the classification along the attributes movement and shape, the seven gestures will be described accordingly.



(a) Continous movement (b) Partitioned movement (c) Path follows curved shape (d) Path follows angular shape

Fig. 2. Taxonomy of gestures along the attributes movement and shape.

As shown in Table 1, every key value has at least been used twice and in combinations with other values that made sense for those particular gestures. Additionally, the "Number of Steps" indicates how many different motions have to be performed in series for this gesture. We created gestures that have multiple steps, which demonstrates the capability of the system to create series of motions. In the following, we give detailed information of each gesture with the aim to transfer the knowledge so that the reader is able to create own gestures. For better readability, the following diagrams are simplified into a 2-dimensional abstraction of the real SMs. The *Circle Gesture* is defined with continuous movement and a curvy shape, see Fig. 3. This gesture seems to be intuitive and is supposed to be used when something has to be rotated. In order to perform this gesture, users have to perform a clockwise circular motion with their arm. The corresponding SM counts to transitions which is at least half of a circle. Start point of the circle can be at any point and due to the self-recovering nature of this particular SM, the recognition works always if the user does not stop circling. Axis and direction of turn can vary in the definition of the SM. The SM shown in Fig. 3. is not parametric.

Table 1. Key values and amount of states per gesture.

	Circle	Shake	Swipe	Hammer	Z	Lever	Ladle
Acc	✔	✔	✔	✔	✔		
Vel			✔	✔			
Grav			✔			✔	✔
||Acc||			✔	✔			
||Vel||			✔	✔	✔		✔
Time		✔	✔				✔
#Steps	5	4	2	2	4	3	6

The *Shaking Gesture* is defined as a continuous movement with an angular shape. This intuitive gesture is an analogy to shake things. An example would be shaking a dice cup. In order to perform this gesture, the executed motion is described by an alternating up and down of the user's arm. The *Shaking Gesture* can either be defined in vertical direction or horizontal direction. The corresponding state machine is depicted in Fig. 4. and shows the definition of this gesture in vertical direction by counting transitions between up and down. The state machine is parametrized in a way that each two sectors

on opposite directions can be used. The *Swipe Gesture* is defined as partitioned gesture following a curvy path, see Fig. 5. This gesture us in analogy to the swipe touch gesture on smartphones that could find use in interactions, where something has to be moved into the direction of the swipe. The gesture can be unintentionally performed in a naturel interaction; therefore, the definition of the state machine is designed in a quite restrictive manner with the use of all key values. In order to avoid the performance of the motion by accident, initially the user has to hold his hand still for around 0,3 seconds, after that he has to move his hand in the wanted direction and hold the speed for a given number of frames. This gesture also detects false alarms which is caused by rotating the wrist, that is done by checking $Grav = Grav_{atStart}$.

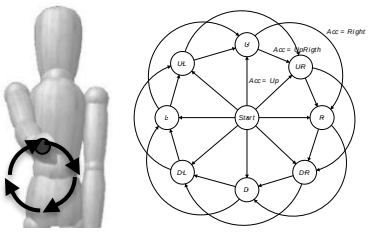


Fig. 3. Motion and SM of Circle Gesture: Every arrow that points to state "L" has the condition Acc = *Left*.

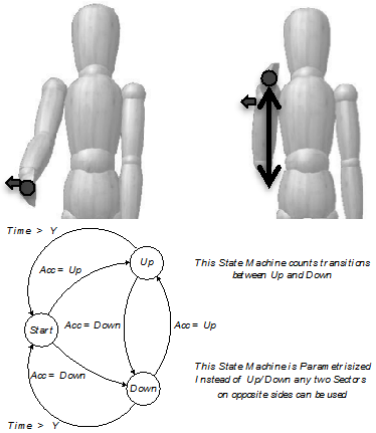


Fig. 4. Motion and state machine of shaking gesture; the gray arrow indicates the direction of the watch.

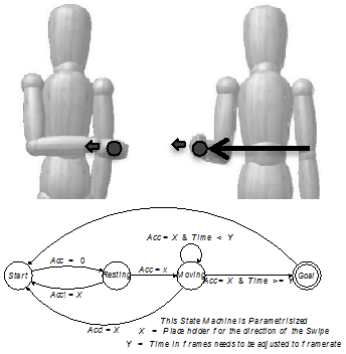


Fig. 5. Motion and state machine of swipe gesture uses all defined key values to avoid unintentionally gesture performance.

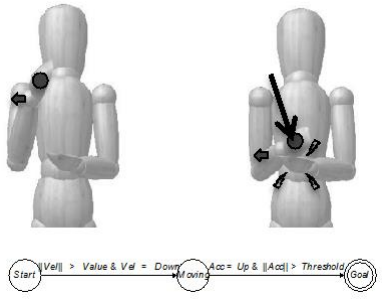


Fig. 6. Motion and state machine of hammer gesture demonstrates the usage of negative acceleration and velocity.

The *Hammer Gesture* follow the definition of a curvy and continuous gesture, see Fig. 6. It is performed by knocking the watch wearing hand in a hammer-swing-like motion onto the other hand. That way the de-acceleration is high enough to make a special

pattern that we detect. It is supposed to be used for pushing buttons, smashing objects, or forging. The *Lever Gesture* was made to evaluate the usefulness of recognizing gestures just by the alignment of the watch to the gravity, see Fig. 7. No other key values are used for the gesture recognition. Therefore, we designed a partitioned gesture following an angular path with two stages in which the watch is rotated in two different directions using the gravity in those directions. First, the watch is rotated along the longitudinal axis and in the second state along the lateral axis. The Ladle Gesture is defined by partitioned movement following a curvy shape, see Fig. 8. The gesture demonstrates the combination of gravity gesture elements and the key values of velocity and time. The motion of the gesture is in analogy to scooping fluid and pouring it into another container. Due to the additional key values, merely rotating the wrist does not trigger the gesture recognition. The *Z Gesture* is defined as a continuous movement along an angular shape and was made to test the limits of the system by combining arbitrary motions into one recognizable gesture, see Fig. 9.

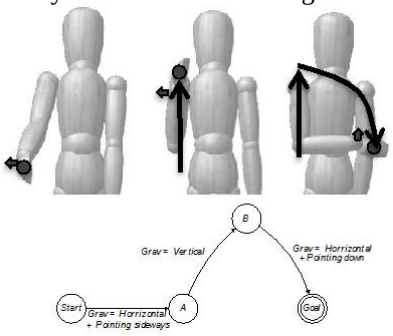


Fig. 7. Motion and state machine of two staged lever gesture only uses gravity as key value.

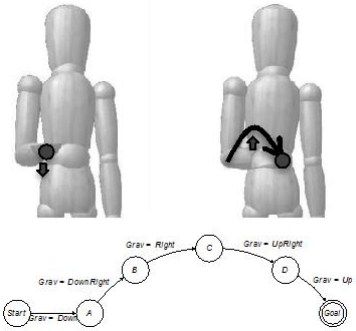


Fig. 8. Motion and state machine of ladle gesture combines gravity with velocity and time values.

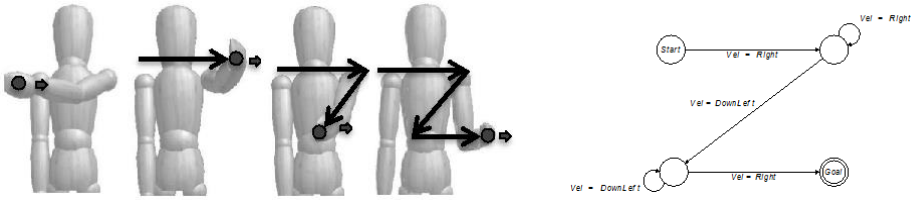


Fig. 9. Motion and state machine of shaking gesture.

6 Discussion and Conclusion

The gestures we provided through the high flexibility of our system are easy to learn, effective, and user show a positive attitude towards using the technology. Primitive gestures as Swipe Left and Swipe Right seem simple, however it is challenging to design those gestures in a way, that they are easy to learn, easy to recognize, but not recognized while performing other gestures. More complex gestures, like Lever, Laddle, or Shaking incorporate less key values and are easier to design. Although, the circle gesture integrates up to nine potential states, not all of those have to be reached making the design and recognition of those gestures easier. Some limitations were

discovered that have to be considered while designing gestures. It can be stated that gestures following continuous movement along angular shapes, like *Z Gesture*, are hard to learn and hard to recognize and should not be used. The reason is that a deceleration of the hand movement is easily recognized as movement into the opposite direction progressing the state machine into the next state too early. Combining arbitrary motions into one recognizable gesture is not possible in any case. Especially, for designing continuous gestures, reversing movements should be avoided. In a user study, we measured the effectiveness, which was measured as accuracy of the gesture recognizer describing the proportion of all measures, correctly classified. An average sensitivity rate of 90.64% for all performed gestures was achieved, with an average specificity rate of 99.46% and average accuracy rate of 98.36%. The best-performed gesture (shaking) had an accuracy of 99.52%, while the "weakest" gesture (swipe right) still had an accuracy of 97.40%. Compared to common technology like other smartwatches or electromyography armbands, the used device in this investigation uses less expensive sensors which can easily lead to inaccurate signals and measurements. Nevertheless, our approach is able to overcome this limitation and it led to satisfying results with low budget devices. Compared to optical tracking systems like 3D cameras, we could already prove in previous work [6] that the usage of smartwatches are promising alternatives to common gesture based interaction technology. Furthermore, our approach is able to identify more diverse gestures and even small movements like rotating the wrist, which would not be recognizable with those optical trackers. In this work, we presented a signal processing approach for enhanced multi-modal interaction interfaces, designed for smart- watches and smartphones for fully immersive environments that enhance the efficiency of interaction in virtual worlds in a natural and intuitive way. This work deals with the replacement of the common touch input gestures with actual body movement gestures. The challenge is described by missing sensor data, precisely gyroscope and magnetometer data, which together with acceleration is used to calculate orientation and motion dynamics of the device. We present a transformation of the giving signal-processing from the smartwatch into arm movement gestures with the use of smoothing algorithms and gesture state machines. Based on six key values and statement sequences we are able to define precise arm movement gestures, exemplary demonstrated on seven different gestures. More gestures are conceivable and easily adoptable with our approach. The findings of the user study prove that the system as described in this work is able to recognize unique, primitive, and even complex gestures in an easy learnable way, while overcoming the missing sensor in low budget technology. The approach used performs quantitatively better results compared to the existing recognizer and allows more divers gestures incorporating different kind of states and key values. The tested gestures covered all key values and with this any kind of possible gesture types. The evaluation shows that complex gestures with many consecutive states are just as well designable as more primitive ones.

Acknowledgments. This research was funded by the German research foundation (DFG) within the IRTG 2057 "Physical Modeling for Virtual Manufacturing Systems and Processes"

References

1. Rupprecht, F., Hamann, B., Weidig, C., Aurich, J. & Ebert, A.: IN2CO - A Visualization Framework for Intuitive Collaboration. In: EuroVis - Short Papers. ACM (2016)
2. Pausch, R., Proffitt, D., & Williams, G.: Quantifying immersion in virtual reality. In Proceedings of the 24th annual conference on computer graphics and interactive techniques, pp. 13–18, ACM (1997)
3. Bryson, S., Feiner, S., Brooks Jr, F., Hubbard, P., Pausch, R. & van Dam, A.: Research frontiers in virtual reality. In: Proceedings of the 21st annual conference on computer, (1994)
4. König, W. A., Rädle, R., & Reiterer, H.: Squidy: a zoomable design environment for natural user interfaces. ACM (2009)
5. Ball, R., North, C. & Bowman, D.: Move to improve: promoting physical navigation to increase user performance with large displays. In: Proceedings of the sigchi conference on human factors in computing systems, pp. 191–200, ACM (2007)
6. Rupprecht, F., Ebert, A., Schneider, A. & Hamann, B.: 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, pp. 2884–2890, ACM (2017)
7. Bergé, L.-P., Serrano, M., Perelman, G., & Dubois, E.: Exploring smartphone-based interaction with overview+ detail interfaces on 3d public displays. In: Proceedings of the 16th international conference on human-computer interaction with mobile devices & services, pp. 125– 134, (2014)
8. Tregillus, S., & Folmer, E.: Vr-step: Walking-inplace using inertial sensing for hands free navigation in mobile vr environments. In: Proceedings of the 2016 chi conference on human factors in computing systems, pp. 1250–1255, ACM (2016)
9. Sharples, S., Cobb, S., Moody, A., & Wilson, J. R.: Virtual reality induced symptoms and effects (vrise): Comparison of head mounted display (hmd), desktop and projection display systems. In: Displays, 29(2), pp. 58–69, (2008)
10. Freeman, E., Brewster, S., & Lantz, V.: Do that, there: An interaction technique for addressing in-air gesture systems. In: Proceedings of the 34th annual conference on human factors in computing systems chi'16, ACM (2016)
11. Wilkinson, G., Kharrufa, A., Hook, J., Pursgrove, B., Wood, et al.: Expressy: Using a wrist-worn inertial measurement unit to add expressiveness to touch-based interactions. In: Proceedings of the conference on human factors in computing systems, ACM (2016)
12. Arefin Shimon, S. S., Lutton, C., Xu, Z., Morrison-Smith, S., Boucher, C., & Ruiz, J.: Exploring nontouchscreen gestures for smartwatches. In: Proceedings of the 2016 chi conference on human factors in computing systems, pp. 3822–3833, ACM (2016),
13. Houben, S., & Marquardt, N.: Watchconnect: A toolkit for prototyping smartwatch-centric cross-device applications. In: Proceedings of the 33rd annual acm conference on human factors in computing systems, pp. 1247–1256, ACM (2015)
14. Mace, D., Gao, W., & Coskun, A. K.: Improving accuracy and practicality of accelerometer-based hand gesture recognition. In: Interacting with Smart Objects, 45, (2013)
15. Mäntyjärvi, J., Kela, J., Korpipää, P., & Kallio, S.: Enabling fast and effortless customisation in accelerometer based gesture interaction. In: Proceedings of the 3rd international conference on mobile and ubiquitous multimedia, pp. 25–31, ACM (2004).
16. Schlömer, T., Poppinga, B., Henze, N., Boll, S.: Gesture recognition with a wii controller. In Proceedings of the 2nd international conference on tangible and embedded interaction, ACM (2008)
17. Chu, H., Huang, S., & Liaw, J.: An acceleration feature-based gesture recognition system. In: International conference on systems, man, and cybernetics, pp. 3807–3812, IEEE (2013)
18. Wagner, F., Schmuki, R., Wagner, T., & Wolstenholme, P.: Modeling software with finite state machines: a practical approach. CRC Press (2006)
19. Wu, J., Pan, G., Zhang, D., Qi, G., Li, S.: Gesture recognition with a 3-d accelerometer. In:

International conference on ubiquitous intelligence and computing, Springer (2009)
20. Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D.: User acceptance of information technology: Toward a unified view. MIS quarterly, pp. 425–478, MIS quarterly (2003)