Microgesture Detection for Remote Interaction with Mobile Devices

Katrin Wolf
BTK University of Art & Design
Berlin, Germany
katrin.wolf@acm.org

Stephan Meyer
Technical University Berlin
Berlin, Germany
st87.meyer@googlemail.com

Sven Mayer
University of Stuttgart
Stuttgart, Germany
sven.mayer@vis.uni-stuttgart.de

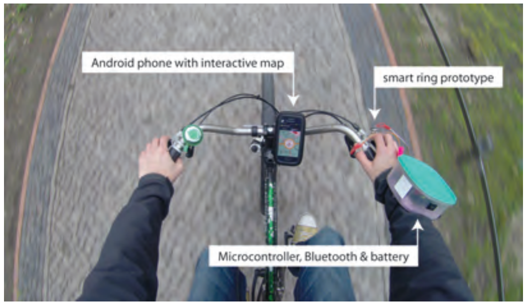
Android phone with interactive map
smart ring prototype
Microcontroller, Bluetooth & battery

Figure 1: Microgesture Interaction for Save Remote Phone Control.

MobileHCI '16 Adjunct,, September 06–09, 2016, Florence, Italy
978-1-4503-4413-5/16/09$15.00
http://dx.doi.org/10.1145/2957265.2961865

Abstract

The rise of smart rings enables for ubiquitous control of computers that are wearable or mobile. We developed a ring interface using a 9 DOF *IMU* for detecting microgestures that can be executed while performing another task that involve hands, e.g. riding a bicycle. For the gesture classification we implemented 4 classifiers that run on the Android operating system without the need of clutch events. In a user study, we compared the success of 4 classifiers in a cycling scenario. We found that *Random Forest* (*RF*) works better for microgesture detection on Android than *Dynamic Time Warping* (*DTW*), *K-Nearest-Neighbor* (*KNN*), and than a *Threshold* (*TH*)-based approach as it has the best detection rate while it runs in real-time on Android. This work shell encourages other researchers to develop further mobile applications for using remote microgesture control in encumbered contexts.

Author Keywords

Microgesture; encumbered contexts; ergonomics; biomechanic; gesture.

ACM Classification Keywords

H.5.2. [User Interfaces]: Input devices and strategies

Background

According to Weiser's vision [15], computers are becoming more and more ubiquitous and users will have the desire to always and everywhere interact with them. Moreover, they will become "invisible" through being embedded into our everyday environment and objects. Today, smartphones are not always immediately accessible, e.g. when the users' hands are busy or if the mobile device is in their pocket. For not requiring to interrupt manual tasks when using our phone, we favor remote microgestures [16] as input technique. They, for example, can be detected with smart rings, even when the hands are busy, e.g. when driving the car or when riding the bicycle, see Figure 1.

Smart rings are a promising form factor for ubiquitous devices as finger rings are already worn in everyday life as jewelry and thus, they are socially accepted [14]. However, available smart rings are often limited to only being an output device, e.g. for showing notification messages. Smart rings that are an input device are still rare [2, 7, 8, 12].

In this work, we are addressing one major challenge in microgesture interaction, which is clutch event free gesture execution, as we aim for fluid and seamless interaction: Clutch events serve to differ between a motion that is intended to be a gesture to control a computer and a natural movement. Avoiding clutch events is important as natural movements that are wrongly interpreted as gesture command cause user frustration and are not acceptable [5, 6].

A common solution to avoid a mismatch between natural movements and gestures are clutch events, a set of in- and output actions, e.g. a button press and release, that labels the gesture so that the computer only analyzes the motions that definitely are meant to be an input command. In real life, as Weiser said, we expect to seamlessly and fluidly interact with ubiquitous computers, which rather leads to avoiding clutch events and to face the challenge of recognizing microgestures out of a continuous data flow [15].

Various technologies that could be embedded into a ring exist and would allow for detecting hand and finger gestures. Optical systems have been used for gesture detection [1, 4, 9, 13], but the handlebars in our cycling scenario would occlude the fingers while they execute the gestures. Capacitive sensors, microphones, and electrodes allow for sensing finger motions under any light conditions and don't suffer from occlusion. They do also not require free hands. For instance, finger gestures have been detected through measuring capacity on a wristband [10], through acoustic signals generated by tapping on the forearm or by forming a fist [3] or through using electromyograms (EMGs) attached at the forearm for hand and finger movement detection [11]. While tapping on the forearm requires both hands, a capacitive wristband and EMG only allow for distinguishing rather rough gestures, but can barely distinguish fine-motor microgestures.

Similar to our approach, the following projects used motion detecting sensors, such as accelerometers and gyroscopes, to recognize microgestures executed with fingers. An accelerometer attached to the fingers enabled for typewriting recognition through tap detection with the *Body Coupled FingeRing* [2]. Scroll gestures have been recognized in *Ubi-Finger* using bend sensors and accelerometers for measuring finger movements [12]. In *Tickle*, a ring interface with an embedded Inertial Measurement Units (IMU) was proposed for detecting tap, release, swipe, and pinch gestures [17]. Jing et al. build a finger ring, which is able to detect one-stroke gestures of the index finger using an accelerometer and detecting the movement speed and direction [7]. *Pingu* a ring interface to control smart environments, was created by Ketabdar et al. to detect mid-air

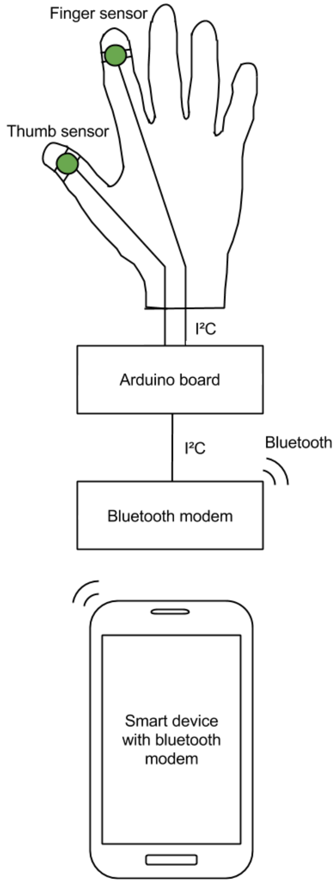

Figure 2: Ring interface connected with a mobile phone.

gestures and taps [8]. To segment the data stream the user touches the proximity sensor of Pingu to generate a clutch event, which our work aims to avoid for better usability.

Although microgestures are promising for remote mobile interaction, previous work on microgesture detection mainly suffers in realistic applicability as it either does not work with current mobile devices or the gesture classification requires clutch events. Thus, previous work does not meet the requirements of natural fluid interaction with ubiquitous mobile devices. In this paper, we present a microgesture ring that enables seamless mobile app control while cycling without interrupting to grasp the handlebars of the bicycle. We implemented and compared 4 microgesture classifiers against each other that run in real time on Android in a pseudo-realistic mobile scenario.

The contribution of this paper is twofold. (1) We present a smart ring device with an embedded IMU to remotely interact with a mobile phone through microgestures. (2) We compare 4 common gesture detection approaches that can run on Android for clutch-free gesture detection, and we show that *Random Forest* is a candidate for detecting microgestures allowing for fluid remote smartphone control.

Ring interface for microgesture interaction

In this section, we explain the ring interface as well as the implementation of the microgesture classification.

The hardware setup consists of two IMUs, containing an MPU-6050 accelerometer and gyroscope each. The IMUs were worn as rings at the index finger and at the thumb. The sensor data was captured with an Arduino pro mini 328 with 8 MHz. We used 8 MHz to keep the amount of data manageable for classifying the data in real time. The data was transferred via Bluetooth to a Samsung S III mini that runs Android. The sensors connected with the Arduino and the Bluetooth modem were communicating using the I^2C protocol. The gyroscope data was used to detect swipe gestures through interpreting the angular velocity around the axis. The swipe directions (left, right, up, and down) correspond with different rotation directions around the sensor axis. Tap gestures are performed with the index finger or thumb. The microgestures were recognized by applying peak detection on the accelerometer data. The schematic setup of our microgesture ring is shown in Figure 2.

For the remote map control, we chose gestures that are commonly used in phone control. Swipe controlled the map panning, while tapping on the front of the handlebar zooms in the map and tapping on the back of the bar zooms out. The gesture recognition is done in three steps: (1) *sensor noise handling* (2) *clutch-free gesture segmentation* (3) *microgesture classification*.

(1) *Sensor noise handling*: The input data stream contains additional sensor noise from finger, hand, and arm motions that are not meant to be a gesture. The noisy data challenges the gesture classification, and we found no common filter that fits our needs to reduce the noise in the input data as digital filters can only reduce noise with defined frequencies. The noise that is emitted from cycling on uneven roads, for example, has no specific frequency. Alternative filter approaches, like Stationary Subspace Analysis (SSA), are computationally too expensive to run on a smartphone. Therefore, we did not reduce the noise in the data, but considered noise as an additional gesture for the gesture classification.

(2) *Clutch-free gesture segmentation*: The unknown start and end points of a gesture in the data stream have to be identified without the help of clutch events. In order to segment the gestures without knowing their start or end time, we estimated start and end by distinguishing between the

hand's motions and its resting positions through filtering the gyroscope signals. We assume that each gesture starts in a resting position, followed by a finger motion, and then ends again in a resting position. If such pattern appears in the input data, the corresponding data will be cropped and classified. Due to intensive pre-tests, we assumed the gestures to be executed within 0.5 s and the highest pace of each gesture to be in the middle of the gesture. Since a new time point is transmitted after $t = 0.33$ s, each gesture consists of $D = \frac{0.5}{0.033} = 15.15$ time vectors. However, since everybody performs the same microgesture a little differently each time, the point of the highest pace will not always be exactly in the middle of the gesture. Therefore, we expanded the data vectors to $D = 20$ data vectors, and thus, each feature vector consists of 10 time points before and after the point of the highest pace. An advantage of this procedure is that it creates feature vectors of the same length. Thus, the classification algorithm does not have to deal with different sized feature vectors.

(3) *Microgesture classification*: After cropping the sensor data stream, we classified the gestures using 4 common and well established algorithms: *Dynamic Time Warping* (*DTW*), *K-Nearest-Neighbor* (*KNN*), *Random Forest* (*RF*), and *Threshold* (*TH*). The main reason for the algorithms choice was that they smoothly and in real time run on Android devices considering the limited computation power of sate-of-the-art mobile devices. If a gesture is interrupted because the user may be not fully concentrated or was interacting with the environment, the gesture might not be recognized depending on gesture execution completeness. Therefore, we implemented a post-classification filter, which smooths the classification results. We count a gesture to be classified if the classifier returns the same classification result in a row for a fixed period. All 4 implemented algorithms run on Android, but they perform differently regarding attributes, which commonly serve to rate the quality of a gesture recognition approach: the *P*-, *R*-, and *F-score*. The *P*-, *R*-, and *F-scores* are values between 0 and 1, whereby a greater value indicates a better recognition result. The *P-scores* measures the ratio between correctly classified gestures and false positives. False positives are gestures that were recognized even though no gesture was executed. The *R-scores* measures the ratio between correctly classified gestures and gestures that were not recognized (false negatives). The *F-scores* is a by a scaling factor α weighted combination of *P*- and *R-scores*. α is a value between 0 and 1 chosen according to whether the *P*- or *R-scores* is more important to the examined scenario.

Without a systematic evaluation of *P*-, *R*-, and *F-scores*, we are not able to propose a classifier that fits well with the requirements for clutch-free microgesture interaction with mobile devices.

Method

For evaluating the classifiers, we compared their performance in a user study regarding their (*P*-, *R*-, and *F-score*) as well as their *classification time*. We conducted an user study where participants performed microgestures (tap and swipe: left, right, up, and down) in a pseudo-realistic mobile scenario by having their hands on rotatable bicycle handlebars, pretending to steer and also to naturally move their hands (reported as no_gesture) from time to time, e.g., for changing a gear or winking a friend. These microgestures where used in our user study because they represent easy and quick to perform gestures, thus these gestures are suitable fore microgestures interaction.

Design

We invited 16 right-handed participants, 6 female and 10 male, aged between 20 and 30 years ($M = 24.6$, $SD =$

3.0), to our study that had a within subject design with the independent variables gesture (no_gesture; swipe: left, right, up, and down; tap: front and back). The dependent variables were the three classification rate parameters *P*-, *R*-, and *F-score* as well as *classification time*.

Apparatus
Our apparatus consisted out of two smart rings and a smartphone, similar to the setup in Figure 1 arranged in a bicycle simulation setup. For the bicycle setup we used a wooden plate to carry the smart phone and the handlebars that were freely movable in the horizontal dimension. An Android app displayed instructions for the gesture training as well as for the gesture execution during the study. The app recorded and classified the data as described in the previous section.

Task
All gestures (swipe: left, right, up, and down; tap at the front and back of the handlebar) as well as no_gesture was performed by all participant with their right/dominant hand while grasping the handlebar through swiping across or through tapping at it.

Procedure
After welcoming participants we asked them to sit down on a chair in front of the bicycle-like setup. Then, we equipped their index finger and thumb with the ring interface. Afterwards, we showed the gestures to the participants and mentioned that they will be asked during the experiment to perform no_gesture through releasing the hand from the handlebar and moving it freely around, e.g. for rearranging the hand on the handlebars, steering or waving. The experiment was split into two phases, the *training phase* and the *classification phase*. The *training phase* started with pressing a start button on the app. Then icons of the gestures that the participants shall execute (five times in a row) appeared on the screen in random order. Before each gesture, 5 s of preparation time, which was displayed as countdown, was given. Through that procedure we roughly predefined the start point for each motion segment that later served for the gesture classification. Each gesture recording lasted 8 s. Then the start button was shown again. That procedure was iteratively repeated until the training sets for each gesture was performed five times. Afterwards no_gesture was continuously recorded over 65 s while the participants rearranged their hand, waved, and rotated the handlebars. During the *classification phase* the participants were asked to execute the gestures in a randomized order. The completely filtered and classified data stream contained the gesture labels for each data point.

Results

During the gestures and no_gesture execution, we recorded gyroscope and accelerometer data. Each participant provided 15 data samples for each gesture, 5 training gestures and 10 test gestures, which are 60 gesture samples per participant and 960 samples in total. In our analyses, we evaluated the 4 classifiers, *DTW*, *KNN*, *RF*, and *TH*, through comparing classification rates and times per gesture. *DTW* uses one template for each gesture to compare the incoming data with. The classification of a sample to a gesture was done with a threshold for the computed distance between sample and template. This threshold was optimized with gradient descent for each participant individually regarding the *F-score* using all recorded trainings samples of that participant. For *KNN*, the Euclidean distance metric between the samples was used. The optimization for the number of neighbors ($k = 2$) was done the same way as *DTW*. During the optimization of *KNN*, it was observed that in all cases the amount of neighbors used for voting was one or two, while the data sets with one neighbor dominated. *RF* consists of many decision trees. Each

	P	**R**	**F**	**time (ms)**
$\bar{x}_{TH}$	$m = .372$	$m = .148$	$m = .224$	0.02
SD$_{TH}$	$\sigma = .042$	$\sigma = .008$	$\sigma = .016$	
$\bar{x}_{RF}$	$m = .936$	$m = .871$	$m = .907$	.98
SD$_{RF}$	$\sigma = .003$	$\sigma = .009$	$\sigma = .003$	
$\bar{x}_{KNN}$	$m = .986$	$m = .775$	$m = .870$	14.16
SD$_{KNN}$	$\sigma = .001$	$\sigma = .020$	$\sigma = .008$	
$\bar{x}_{DTW}$	$m = .389$	$m = .297$	$m = .348$	186.02
SD$_{DTW}$	$\sigma = .149$	$\sigma = .036$	$\sigma = .101$	

Table 1: Classifier comparison by mean ($\bar{x}$) and the SD of the *P*-, *R*-, and *F-score* as well as the classification time in milliseconds.

Results	***TH-RF***	***TH-KNN***	***TH-DTW***
$H_0(P)$	< 0.001*	< 0.001*	0.871
$H_0(R)$	< 0.001*	< 0.001*	0.009
$H_0(F)$	< 0.001*	< 0.001*	0.160
Results	***RF-KNN***	***RF-DTW***	***KNN-DTW***
$H_0(P)$	0.004*	< 0.001*	< 0.001*
$H_0(R)$	0.037*	< 0.001*	< 0.001*
$H_0(F)$	0.231	< 0.001*	< 0.001*

Table 2: Bonferroni-corrected p-values for pairwise comparisons of the classification algorithms.

decision tree votes for one of the trained classes the gesture will be classified to. The class with the majority of votes is picked as the result class. Furthermore, feature selection was used to determine, which feature of each data point shall be used by a decision tree during the classification. The threshold of the *TH* algorithm was determined during the training phase. During each gesture, the highest velocity and direction of the gesture was determined for all five training samples. The mean of these velocities was used as the threshold for each gesture during the classification phase.

To test if our implemented algorithms fulfill the requirements of classifying microgestures in mobile scenarios, we compared their classification rate (*P*-, *R*-, and *F-score*) and the *classification time* as shown in Table 1.

The gesture classification times differed a lot between the algorithms, and *TH* was with 0.02 s the fastest. *RF* was 49 times slower than *TH*, *KNN* was 708 times slower, and *DTW* 9301 times slower than *TH*. A one-way *ANOVA* indicated that the choice of the classification algorithms significantly influenced the classification rates (*P*-, *R*-, and *F-score*). The classification algorithm effected the *P-score* ($F_{3,60} = 37.85, p < 0.001$), the *R-score* ($F_{3,60} = 62.39$, $p < 0.001$), and the *F-score* ($F_{3,60} = 39.64, p < 0.001$). Bonferroni-corrected post-hoc comparisons yielded significant differences between all scores of *TH* vs. *RF* ($p < 0.001$), *TH* vs. *KNN* ($p < 0.001$), and *KNN* vs. *DTW* ($p < 0.001$). Comparing *RF* vs. *KNN* indicated significant differences for the *P-score* ($p = 0.004$) and the *R-score* ($p = 0.037$), but not for the *F-score* ($p = 0.231$). We did not find any significant differences between *TH* vs. *DTW* ($p > 0.05$), as shown in Table 2 as well as in Figure 3.

In summary, *Random Forest* resulted in the highest *F-score* and therefore, it led to the best recognition rate with a comparatively small amount of false positive gestures due to the choice of $\alpha = 0.7$, and a relatively small variance. *K-Nearest-Neighbor* has the second highest *F-score*, followed by *Dynamic Time Warping*, while *Threshold* has the lowest score. Moreover, the precision P (that was raised by the smoothing step at the end of the classification) is in all cases higher than the Recall R. Finally, *DTW* has a smaller precision value P than we would have expected due to its commonly established usage.

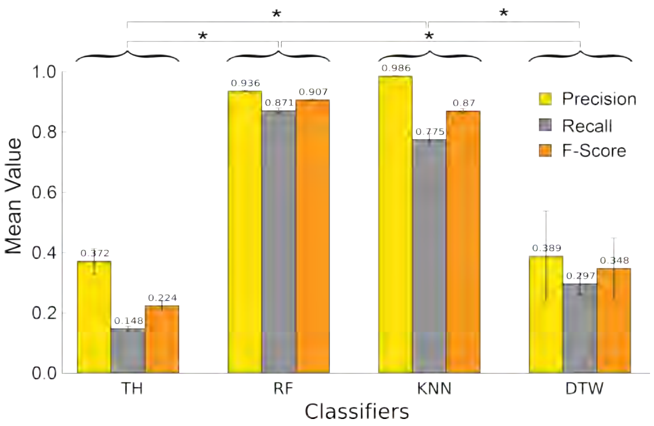

Figure 3: Recognition means including standard error represented by the error bars, significant differences are marked with a *.

Discussion & Conclusion

In this paper, we introduced a ring interface that detects microgestures as input for smartphones. We showed that under the limitations of smart phones, *Random Forest* works best for microgesture classification compared with *K-Nearest-Neighbor*, *Dynamic Time Warping*, and a *Threshold* approach.

In our classifier implementation, we addressed the following challenges of using microgestures for remotely controlling mobile devices: The classifiers run on an Android phone. No gesture labeling or so called clutch events are used for gesture segmentation. The gesture data stream contains also natural movements to challenge the clutch-free gesture classification. Under the 4 tested classification approaches, *Random Forest* shows a high classification rate ($F_{RF} = 0.907$), and it classifies a gesture in 0.98 ms only.

Through demonstrating that clutch-free microgesture recognition is possible on Android devices and even if the signals contain natural movements, we show that ring interfaces not only can be used as micro-displays like their currently are, they also could be utilized to be a ubiquitous remote controller for all kinds of devices, such as mobile phones.

References

[1] Gilles Bailly, Jörg Müller, Michael Rohs, Daniel Wigdor, and Sven Kratz. 2012. ShoeSense: A New Perspective on Gestural Interaction and Wearable Applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 1239–1248. DOI: http://dx.doi.org/10.1145/2207676.2208576

[2] Masaaki Fukumoto and Yoshinobu Tonomura. 1997. "Body Coupled FingerRing": Wireless Wearable Keyboard. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '97)*. ACM, New York, NY, USA, 147–154. DOI: http://dx.doi.org/10.1145/258549.258636

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

[4] Bruce Howard and Susie Howard. 2001. Lightglove: Wrist-Worn Virtual Typing and Pointing. In *Proceedings of the 5th IEEE International Symposium on Wearable Computers (ISWC '01)*. IEEE Computer Society, Washington, DC, USA, 172–173. http://dl.acm.org/citation.cfm?id=580581.856559

[5] Howell Istance, Richard Bates, Aulikki Hyrskykari, and Stephen Vickers. 2008. Snap Clutch, a Moded Approach to Solving the Midas Touch Problem. In *Pro-*

ceedings of the 2008 Symposium on Eye Tracking Research; Applications (ETRA '08). ACM, New York, NY, USA, 221–228. DOI: http://dx.doi.org/10.1145/1344471.1344523

[6] Robert J. K. Jacob. 1990. What You Look at is What You Get: Eye Movement-based Interaction Techniques. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '90)*. ACM, New York, NY, USA, 11–18. DOI: http://dx.doi.org/10.1145/97243.97246

[7] Lei Jing, Zixue Cheng, and Wang Junbo. 2011. A recognition method for one-stroke finger gestures using a MEMS 3D accelerometer. IEICE transactions on information and systems 94.5, 1062–1072.

[8] Hamed Ketabdar, Peyman Moghadam, and Mehran Roshandel. 2012. Pingu: A New Miniature Wearable Device for Ubiquitous Computing Environments. *2014 Eighth International Conference on Complex, Intelligent and Software Intensive Systems* 0 (2012), 502–506. DOI: http://dx.doi.org/10.1109/CISIS.2012.123

[9] David Kim, Otmar Hilliges, Shahram Izadi, Alex D. Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. 2012. 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 (UIST '12)*. ACM, New York, NY, USA, 167–176. DOI: http://dx.doi.org/10.1145/2380116.2380139

[10] Jun Rekimoto. 2001. GestureWrist and GesturePad: Unobtrusive Wearable Interaction Devices. In *Proceedings of the 5th IEEE International Symposium on Wearable Computers (ISWC '01)*. IEEE Computer Society, Washington, DC, USA, 21–. http://dl.acm.org/citation.cfm?id=580581.856565

[11] T. Scott Saponas. 2009. Enabling Always-available Input: Through On-body Interfaces. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems (CHI EA '09)*. ACM, New York, NY, USA, 3117–3120. DOI: http://dx.doi.org/10.1145/1520340.1520441

[12] Koji Tsukadaa and Michiaki Yasumurab. 2001. Ubi-finger: Gesture input device for mobile use. In *Proceedings of Ubicomp*. 11.

[13] A. Vardy, J. Robinson, and Li-Te Cheng. 1999. The WristCam as input device. In *Wearable Computers, 1999. Digest of Papers. The Third International Symposium on*. 199–202. DOI: http://dx.doi.org/10.1109/ISWC.1999.806928

[14] Katia Vega and Hugo Fuks. 2013. Beauty Technology As an Interactive Computing Platform. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces (ITS '13)*. ACM, New York, NY, USA, 357–360. DOI: http://dx.doi.org/10.1145/2512349.2512399

[15] Mark Weiser. 1991. The computer for the 21st century. *Scientific american* 265, 3 (1991), 94–104.

[16] Katrin Wolf, Anja Naumann, Michael Rohs, and Jörg Müller. 2011. Taxonomy of Microinteractions: Defining Microgestures Based on Ergonomic and Scenario-dependent Requirements. In *Proceedings of the 13th IFIP TC 13 International Conference on Human-computer Interaction - Volume Part I (INTERACT'11)*. Springer-Verlag, Berlin, Heidelberg, 559–575. http://dl.acm.org/citation.cfm?id=2042053.2042111

[17] Katrin Wolf, Robert Schleicher, Sven Kratz, and Michael Rohs. 2013. Tickle: A Surface-independent Interaction Technique for Grasp Interfaces. In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction (TEI '13)*. ACM, New York, NY, USA, 185–192. DOI: http://dx.doi.org/10.1145/2460625.2460654