# Performance analysis of an HMM-based gesture recognition using a wristwatch device

Roman Amstutz*, Oliver Amft*†, Brian French‡, Asim Smailagic‡, Dan Siewiorek‡, Gerhard Tröster*
*Wearable Computing Lab., ETH Zurich, CH-8092 Zurich, email: {amft,troester}@ife.ee.ethz.ch
†Signal Processing Systems, TU Eindhoven, NL-5600 MB Eindhoven, email: amft@tue.nl
‡Carnegie Mellon University, Pittsburgh, PA 15213, USA, email: bfrench@andrew.cmu.edu, {asim,dan}@cs.cmu.edu

***Abstract*—Interaction with mobile devices that are intended for everyday use is challenging since such systems are continuously optimized towards small outlines. Watches are a particularly critical as display size, processing capabilities, and weight are tightly constraint.**

**This work presents a watch device with an integrated gesture recognition interface. We report the resource-optimized implementation of our algorithmic solution on the watch and demonstrate that the recognition approach is feasible for such constraint devoices. The system is wearable during everyday activities and was evaluated with eight users to complete questionnaires through intuitive one-hand movements.**

**We developed a procedure to spot and classify input gestures from continuous acceleration data acquired by the watch. The recognition procedure is based on hidden Markov models (HMM) and was fully implemented on a watch. The algorithm achieved an average recall of 79% at 93% precision in recognizing the relevant gestures. The watch implementation of continuous gesture spotting showed a delay below 3 ms for feature computation, Viterbi path processing, and final classification at less than 4 KB memory usage.**

***Index Terms*—Event spotting, algorithm implementation, intelligent wristwatch, recognition performance, mobile interaction, eWatch**

## I. Introduction

Mobile and wearable devices are continuously optimized towards a small outline. At the same time the number of functions in these devices continuous to increase. While this development is clearly beneficial for the ubiquity of mobile and wearable systems, e.g. for using the systems during daily activities, it hampers interaction. System functionalities require more complex forms of interaction and longer interaction times. Classic interaction modalities, such as buttons or touchpads can even constrain further size reductions. This raises the need for alternative interaction approaches.

Watches are a particularly affected by these issues as form factor, display size, processing capabilities, and weight are tightly constraint in order to maintain their usability. Nevertheless the benefit of watches for users to obtain information is excellent. In particular, watches can provide details on personal state, such as emotions and context of the wearer, which are relevant in medical and behavioral monitoring and assistance applications. Moreover, their ubiquitous use during everyday life can retain privacy of the user, as the assistance functions are hidden in a device which is frequently worn as part of the outfit. Primarily button-based interfaces have been devised for watch devices. While these interfaces are well-established, buttons require close attention, two-arm operation (one to hold the watch, the second to operate the buttons), and good motor skills in fingers and both arms for successful operation. For example patients after stroke do not fulfil these requirements, as they may be capable of using only one arm actively.

We aim to advance watch-based interaction concepts, by developing (1) a watch device that can be conveniently worn, and (2) a simple method to interact with it using hand gestures. These properties permit to use a watch for an assistance systems during everyday life, e.g. to control appliances in stroke patients and support their re-integration into their pre-stroke life. Moreover, it permits to gain feedback from the patients on their integration process by implementing questionnaires into a watch. It may even allow to use the system under special working conditions, such as for an assembly worker who needs at least one arm to operate machinery, thus preventing many classic interaction options. In this paper, we introduce a watch system that relies on a gesture interface to acquire responses for questionnaires. We have selected a questionnaire application to demonstrate the functionality of our watch device, since a questionnaire requires many interaction steps. Hence it demonstrates how even complex information could be input into a assistance system.

The challenges of this approach are the design and implementation of a gesture recognition procedure onto a resource-limited watch device. The recognition of questionnaire and menu control gestures has to be performed on continuously arriving sensor data. In particular, the spotting of relevant gestures must be robust against arbitrary unrelated motions, in order to not accidently answer the questionnaire. Moreover, the system should be responsive to the wearer's gestures, hence the processing delay must be low. Finally, the gestures shall be correctly identified among the set of relevant gestures.

While gesture recognition algorithms have been broadly studied in the literature (see Section II below), implementation report that detail system performance are rare and limited to high-resource mobile devices, such as mobile phones. In this regard, this paper makes the following contributions:

1) We present a watch system, the eWatch, for questionnaire entry and describe its design choices in Section III. The selection of the control gestures is discussed and the eWatch architecture summarized.
2) We introduce an hidden Markov model (HMM)-based recognition procedure to detect control gestures in con-

tinuous data. The recognition procedure was evaluated in a study with eight users. Based on this evaluation, we present a quantitative analysis of the recognition performance. The recognition procedure and evaluation results are summarized in Section IV.

3) We present our eWatch implementation and analysis results of the gesture recognition in Section V. We evaluated the eWatch implementation regarding its processing requirements and constraints, such as the recognition delay and computational requirements.

## II. Related work

Alternate input concepts for system menus and questionnaire assessments have been investigated to reduce the user's efforts or adapt to a handicap in mobile systems. PDAs or similar electronic systems are frequently used as replacements for the classic paper-based self-reports and were found to provide equivalent results [1]. In cognitive or visually impaired populations voice recording were used however found to be ambiguous [2]. While sound- or vision-based approaches may well suit handicapped users, these typically require extensive processing and communication resources. Both are extremely limited in a watch device.

Wristwatches have been investigated as processing units in wearable computing as well as for many commercial applications besides time measurement. A pioneering work was the IBM Linux Watch [3]. With the introduction of Microsoft's Smart Personal Object Technology (SPOT) [4], consumer watches became broadcast news receivers. Similarly, wristwatches can be used as a mobile phone [5] or for GPS-navigation [6]. Besides the frequent button-based control, wristwatches have been equipped with touch-sensitive displays [7] to improve interaction. Gesture interfaces for watches remain an unexplored option.

Hand and arm gesture-control is a prospective interaction method that offers convenient properties to manage mobile and wearable systems. Despite its advantages, commercial gesture-controlled mobile or wearable devices remain rare. This is explained by complex sensing and command recognition required to handle gesture input. Natural hand and arm movements are variable in their spatial and temporal execution, requiring more tolerant pattern recognition solutions. Hidden Markov models (HMMs) have been successfully applied for this purpose. However, only few implementation approaches for mobile devices have been reported, as detailed below.

Several algorithm approaches have been made to recognize gestures in continuous sensor data. Most recognition solutions are based on HMMs, since HMMs can cope with temporal variations in the gesture duration. Moreover the recognition procedure needs to discriminate actually conducted gestures from non-gesture patterns (other, arbitrary movements). Lee and Kim [8] recognized gestures to control a slide show. In their approach non-gesture patterns were filtered using a threshold model. Deng and Tsui [9] investigated two-dimensional trajectories of Arabic numbers. An accumulation score was used to find gesture end-points and the Viterbi algorithm to trace the beginning. A fixed threshold and a rule set was used to evaluate the final recognition result. Junker et al. [10], [11] deployed a two-stage recognition procedure. In the first stage gesture candidates were identified from the continuous data using a Feature Similarity Search, in the second stage HMMs where used to remove false positives from the results. The approach was evaluated on different natural gestures. While these works have focused to develop recognition solutions, none of them provided relevant implementation details and analysis results for watches or similar resource-restricted watch devices.

Among the few HMM-based recognition implementations that have been analyzed for gesture recognition in mobile systems, Kallio et al. [12] reported an average processing delay of 8.3 ms for a discrete 5-state HMM. However, their investigation only addressed HMM classification and did not consider a complete procedure for continuous recognition, covering spotting and classification. Niezen and Hancke [13] presented an recognition approach on a mobile phone device. While they report on the implementation, no performance assessment of the recognition nor resource consumption analysis was made. Kim et al. [14] introduced a watch-like device to recognize hand gestures using an array of infrared proximity sensors. The system requires both arms/hands to operate the device. While it achieves a similar accuracy compared to the acceleration-based approach taken in this work, the authors did not investigate implementation aspects and resource usage.

Our contribution pioneers in analyzing implementation aspects of the recognition and in demonstrating a fully functional gesture-controlled solution. We regard this as an essential step towards establishing gesture recognition as an alternate interaction modality for watches and similar embedded devices with minimal processing and memory resources.

## III. Watch-based questionnaire system

### A. eWatch architecture

The eWatch consists of a CPU, sensors, power control, notification mechanisms and wireless communication. The CPU is a micro-controller of the ARM7TDMI processor family without float-point unit, running with up to 80 MHz. The system architecture is further detailed in [15]. In this work a MEMS 3-axes accelerometer with a sampling rate of 20 Hz was used to record acceleration of the wearer's arm.

### B. eWatch questionnaire application

The questionnaire concept applied in this work is based on a periodic polling of the wearer on his or her current state. We foresee that the system will be used for several interviews during a day. To collect representative results, the interviews have to be as unobtrusive as possible. The eWatch allows its wearers to answer the questionnaire with minimal influencing their daily routine. For example, in one application scenario the eWatch-based interview starts automatically every 45 minutes. The user is notified about the start of an interview by a vibration alarm. The questionnaire comprises a set of questions that are sequentially answered. One question at a

time is shown on the screen, each having two or four response options. An example screenshot is depicted in Figure 1. The response options are displayed as boxes in a vertical order on the screen. The intended response is selected by scrolling up and down between the different answer boxes. After selecting the response, the screen displays the next question.


Fig. 1. Questionnaire application running on the eWatch. Left: Initial state, right: state after scrolling up.

Figure 2 shows the eWatch worn while using gesture control. Figure 3 shows the state diagram of the watch-adapted questionnaire using gesture-control. The *scroll* gestures are used to toggle between the different answers, while *select* gesture logs the selected response. To prevent the system from immediate switching to the next question when the select gesture was accidentally recognized, a *confirm state* was introduced. However the gesture recognition works robustly, such that an erroneous recognition of a select gesture occurs rarely. Nevertheless, this additional confirmation feature prevents wrong user inputs as well.

To attain a sufficient user acceptance, gestures used to navigate through the questionnaire must be convenient to perform and easy to learn. Similarly, gesture patterns shall be distinctive to ensure good recognition performance. We expect the performance to be highly dependent on how familiar a user is with the system. Since the performance can be improved by user training, our main focus was the design of a highly precise gesture recognition algorithm. Based on a preliminary analysis of gesture patterns for different arm motions, three gestures were chosen (see Figure 2).

Scrolling up is initiated by an arm movement away from the body with the palm facing ground. Scrolling down is similar, but the arm movement is towards the body. The select gesture is executed in perpendicular direction to the plane of both scrolling gestures. It consists of two subsequent arm lifts. With this selection the gestures generate characteristic movement patterns, while being comfortable to execute.

## IV. HMM-BASED GESTURE RECOGNITION

### A. Online recognition method

A three-stage recognition procedure was developed to detect and identify control gestures from a continuous stream of 3D-acceleration data recorded using the eWatch. The challenge to spot gestures in continuous acceleration data is related to the high chance of observing arbitrary other movements (NULL class) that could be confused with the interested gestures. Our approach specifically addresses this challenge by integrating a NULL class rejection in the procedure.

Figure 4 illustrates the recognition procedure using the recorded acceleration data. Firstly, an informative feature was extracted from the recorded 3D-acceleration data. Figure 5 shows the acceleration of the z-axis and the corresponding derivative exemplary. The dominant acceleration axis was determined as axis with the largest amplitude variation within the last five sampling points. We used this derivative as feature to characterize every sampling point. Subsequently, a sliding window of fixed size (30 samples) was shifted over the feature data with a step size of 5 samples.

In a second stage, the Viterbi algorithm [16] was applied to detect begin and end samples of potential gestures. For this purpose, each gesture type was modeled by an individual left-right discrete HMM. Six states were used for scroll gestures, nine states for the “Select” gesture. A code-book of 13 symbols was used to represent the derivative amplitude in strong/low increase/decrease for all acceleration axes and calm, for small amplitudes.

An initial analysis showed that a period without movement preceded and followed each gesture. This period occurred naturally, when the user read the next question from the screen or confirmed the completion of the current one. Thus, the first and last states of all models were designed to represent small acceleration variations.

Figure 6 shows a simplified model designed for the “Select” gesture. The derivative amplitude, originally represented in 13 symbols, was reduced for this illustration. The states correspond to different phases in the acceleration pattern plotted in Figure 5. The states 2, 4 and 7 represent the phases of increasing acceleration while 3, 6 and 8 model a decrease. State 5 characterizes a period of arbitrary acceleration values in the center of the gesture pattern. The models for scroll gestures were derived accordingly. Arbitrary further gestures could be modelled with this approach.

The Viterbi algorithm was deployed to calculate the most probable state transition sequence for the current window. This state transition sequence is subsequently referenced as Viterbi path. We use the Viterbi path to detect the begin and end of a potential gesture. Here, the end detected if the end state in an HMM was reached in the current window. Using this end-point, the corresponding gesture begin was determined using the HMM. The HMMs were manually designed and parameterized by reviewing a training data set that was independently recorded before the algorithm evaluation. The manual design was chosen since the gesture set does not correspond to gestures frequently occurring in daily life. Moreover the set can be conveniently described using states as summarized above. In contrast, users would perform the gestures with large variances between the executions without previous training.

The third stage of the procedure classified a potential gesture detected in stage two. To determine whether to retain or discard a gesture, the likelihood value, as obtained from

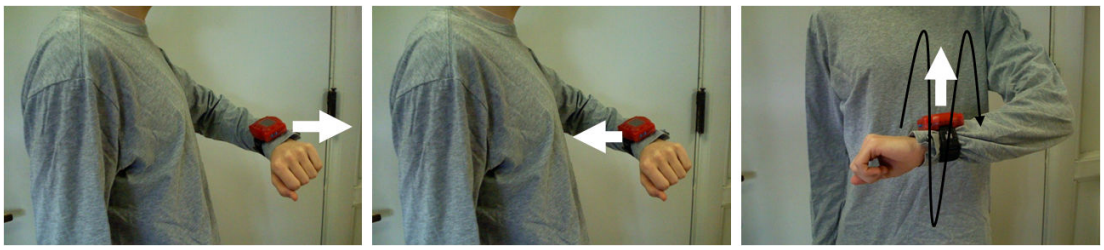

Fig. 2. Gestures used to control the watch interface: Scroll-up, scroll-down, and select gestures

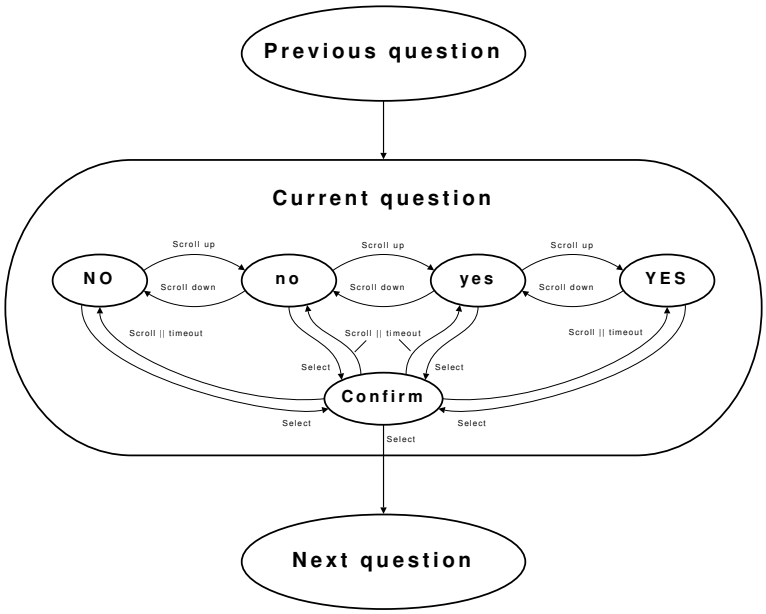


Fig. 3. Questionnaire state diagram as it has been adapted to the watch. Gestures Scroll up, Scroll down, Select have been devised to navigate between the states.

the Viterbi path computation, was compared to an adaptive threshold. The threshold was derived using a threshold model, as described by Lee and Kim [8]. If more than one gesture was detected in one window, the one with larger likelihood was retained.

### *B. Dataset and evaluation procedure*

The recognition procedure was designed and parameterized using an initial dataset collected from five users. Before the recording, these users received a brief description and demonstration of the control gestures. The users performed 20 training gestures of each category. An observer labeled the acceleration data.

To derive the gesture models, the acceleration data was manually segmented into a series of gesture phases. HMM states were designed to correspond with these phases.

To subsequently validate the designed models, the recognition procedure was tested using acceleration data from eight further users. Before the actual test data collection, the users were allowed to practice the gestures. For this purpose, a Matlab host-based simulation of the algorithm was fed with the acceleration data from the eWatch. A Bluetooth link provided online visual feedback on the recognition result of practiced

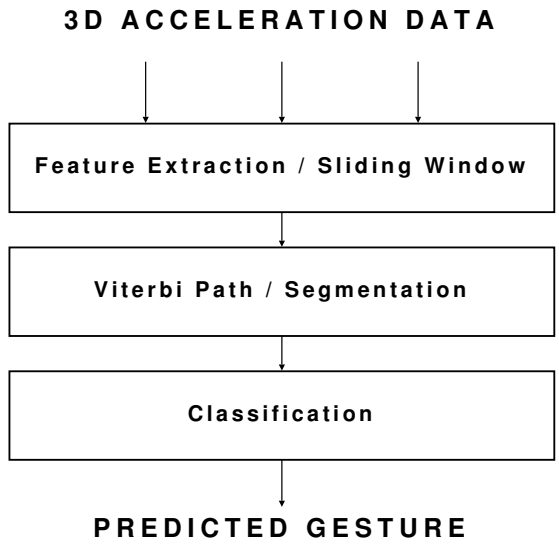


Fig. 4. Recognition procedure developed to spot and classify gestures in continuous acceleration sensor data. The Viterbi path stage allows to reduce the search complexity for the subsequent HMM-based classification.

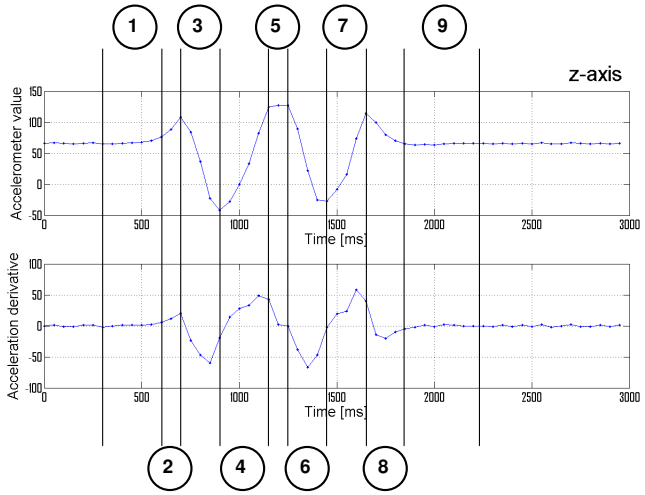


Fig. 5. Example z-axis acceleration data (from the eWatch device) and corresponding derivative recorded for the "Select" gesture. The sections indicated with 1–9 correspond to states of the "Select" gesture model illustrated in Fig. 6.

gestures. The training lasted for about 5 minutes and helped the users to accommodate with the watch and performing the gestures. Although this practice phase was short, users quickly adopted the gestures during this time. In a related analysis users rated physical effort constantly while the focus on the gestures decreased when performing multiple gesture repetitions [17]. This indicates that such a practicing step is a useful approach for specifically designed gestures, as they are used here. This training does not hamper generalization of our results since we expect that users of the watch would obtain a similar proficiency after a few minutes of practice.

After the practice session, users performed 608 gestures, including all relevant gestures as well as arbitrary motions such as drinking and handling a cell phone. Moreover, the dataset included the motions needed to handle the watch, e.g. moving it to observe the screen. This data served to study the risk of false detections. The acceleration data was recorded from the eWatch for training and test dataset.

The recognition performance was analyzed using the metrics *Precision* and *Recall* [11]. *Relevant gestures* correspond to the actually conducted and manually annotated gestures, *retrieved gestures* are all gestures returned by the procedure and *recognized gestures* represent the correctly returned gestures. A summary and visualization of the different sets can be found in [11].

To identify a gesture as correctly returned by the procedure, a time-domain overlap check between the labeled and retrieved gestures was used.

### C. Recognition performance results

Table I summarizes the recognition performance obtained for the test dataset. A average recall performance of 79% was achieved at a precision of 93%. For all gestures, the procedure achieved a high precision, at the expense of a lower recall. Especially, the "Select" gesture was optimized to obtain a high precision, since this gesture allows to complete a question. The remaining errors indicate the recognition complexity. In our dataset arbitrary motions (NULL class) prevent the ideal result of identifying all relevant gestures and omit all arbitrary ones. However, it can be expected that in a real application arbitrary motions are minimal during interaction times. Additional activation gestures, such as entering the watch menu or activating the questionnaire may be used to further reduce the risk of accidental misinterpretations.

Table II shows the recognition confusion matrix. Here, the procedure obtained a low number of confusions among the relevant gestures ($\leq$6 gestures were confused). Other, arbitrary movements were only 11 times identified as a relevant gesture. This confirms the robust recognition result.

| **Metric** | **Gestures** | | |
|---|---|---|---|
| | Scroll-up | Scroll-down | Select |
| Relevant | 170 | 149 | 160 |
| Recognized | 144 | 113 | 120 |
| Insertions | 10 | 17 | 1 |
| Deletions | 26 | 36 | 40 |
| **Recall** | **0.85** | **0.76** | **0.75** |
| **Precision** | **0.94** | **0.87** | **0.99** |

TABLE I
PERFORMANCE EVALUATION FOR CONTINUOUS CONTROL GESTURE RECOGNITION.

| **True gestures** | **Predicted gestures** | | |
|---|---|---|---|
| | Scroll-up | Scroll-down | Select |
| Scroll-up | 144 | 6 | 0 |
| Scroll-down | 3 | 113 | 0 |
| Select | 0 | 0 | 120 |
| Arbitrary motion | 7 | 11 | 1 |

TABLE II
CONFUSION MATRIX OF THE CONTROL GESTURE RECOGNITION ON THE TEST DATASET.

## V. IMPLEMENTATION ANALYSIS

### A. Implementation details

Our implementation of the gesture recognition procedure uses the eWatch system code to manage hardware components, such as sensors, timers and the display. For the investigations made in this work a CPU clock of 65 MHz was used.

To minimize sampling jitter, the eWatch uses an interrupt-based sampling of the acceleration sensor. In the interrupt handler, the ADC value is read and stored in a memory buffer. The gesture recognition implementation runs at an acceleration sensor sampling rate of 20 Hz. To reduce CPU load, the gesture recognition procedure is launched at a rate of 4 Hz only. This rate is aligned to the sliding window with a step size of five samples.

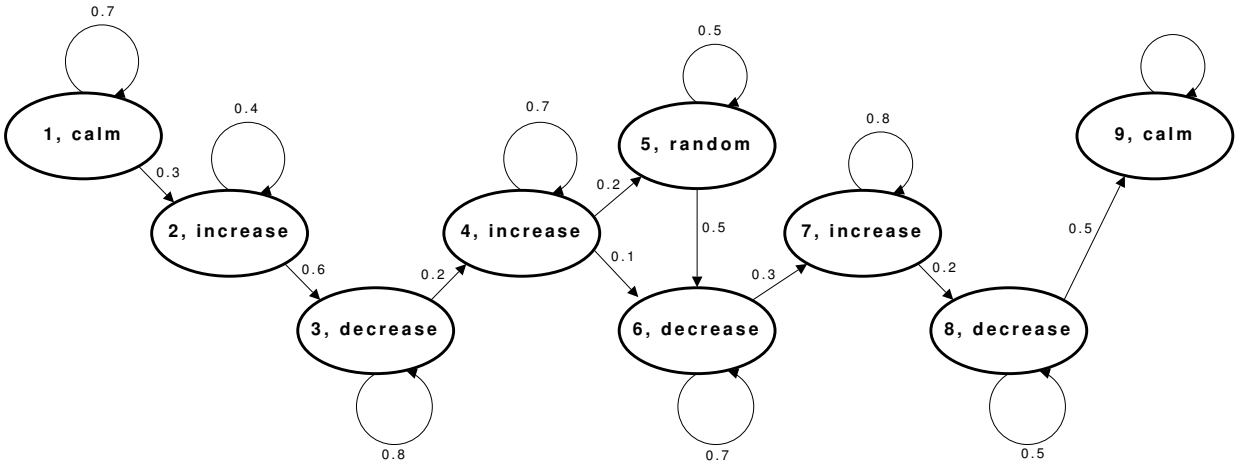


Fig. 6. Illustration of the HMM state transitions of the "Select" gesture with state transition probabilities (simplified).

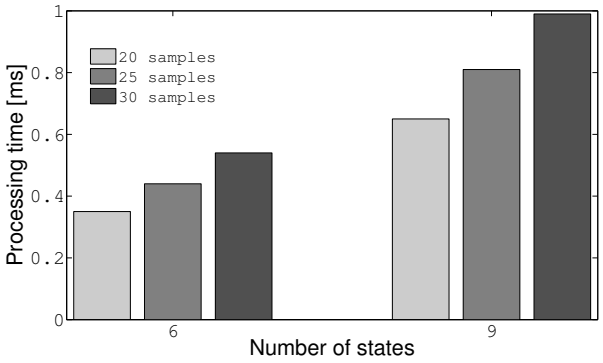


Fig. 7. Viterbi path processing time in relation to sliding window size and number of HMM states.

The first stage of the recognition procedure was implemented using conditional execution, branching and addition operations. In the second stage, the Viterbi path calculation was required. Since the eWatch CPU does not provide float-point unit, all likelihood values were stored as integers. By using logarithmic values of the discrete probabilistic distributions, the computation of the Viterbi path was reduced to integer additions.

The analytical description of a HMM consists of two matrices (observation and transition matrix) and one vector (prior probabilities). The parameter set was stored in memory for three gesture models and a null-class model. In total, 8 matrices and 4 vectors of integer values, with a memory footprint of $\sim$2 KB, were statically allocated. Another 1.75 KB were used to store temporary values during the calculations. These buffers were statically allocated as well, to prevent time consuming dynamic memory handling during the processing.

### *B. Processing performance results*

Table III shows the average processing time for the different stages of our recognition procedure. Processing times were determined for the eWatch implementation (C code) and the host-PC (Matlab). The numbers indicate the time needed to compute all three gesture models and the threshold model within the sliding window (size: 30 samples, accelerometer sampling rate: 20 Hz).

The calculations of the second stage (Viterbi path) take about 97% of the total processing time. On average the gesture recognition takes 2.7 ms on the eWatch. Since the algorithm runs at 4 Hz, only 1% of the total CPU time is needed. This result confirms the applicability of the recognition procedure for small wearable devices. Moreover, the processing performance results show that, the eWatch implementation is twice as fast as the host-based Matlab implementation. This result clearly shows that the eWatch-implementation gained from the conversion and optimization of directly executable code.

| Recognition stage | **eWatch** (C code) | **host-PC** (Matlab) |
|---|---|---|
| Feature extraction | 0.1 ms | 0.2 ms |
| Viterbi path | 2.6 ms | 5.2 ms |
| Classification | <0.1 ms | <0.1 ms |
| **Total** | **2.7 ms** | **5.4 ms** |

TABLE III
PROCESSING PERFORMANCE ANALYSIS ON THE EWATCH AND ON A HOST-PC (MATLAB IMPLEMENTATION).

Figure 7 shows the relation of Viterbi path processing time with the number of HMM states and the sliding window size. This evaluation was performed using the eWatch implementation to process one model. The processing time increased linearly with the sliding window size and quadratically with the number of HMM states. The analysis showed a processing time of $< 0.6\,ms$ for a 6-state HMM and 1 ms for a 9-state HMM, both at a sliding window size of 30 samples.

The results are confirmed by the theoretical computational complexity of the Viterbi algorithm. The complexity

is $O(n^2m)$, where n is the number of HMM states and m is the window size.

As summarized in the section on related works before, Kallio et al. [12] reported an average processing delay of 8.3 ms for a discrete 5-state HMM. However, the authors did not report on their implementation details. Moreover, their results relate to a classification only, not a fully continuous recognition. Hence, the results are not directly comparable. Nevertheless our performance details cover all functional elements. Consequently the results indicate a sufficient performance for an watch or embedded system implementation with constraint resources.

## VI. Discussion and conclusion

In this work, we presented a novel wearable questionnaire assessment system based on the eWatch device. The system permits patient support and behavioral monitoring of their wearers. Using a gesture-controlled interface, the watch-based questionnaire can be completed with one hand only. We believe that this system has large benefits for monitoring cognitive or emotional state, control appliances, as well as in many further assistance applications.

We presented in this paper the watch-based questionnaire concept and a recognition procedure to acquire gestures. We confirmed the feasibility of a gesture-controlled wristworn watch by evaluating the recognition performance with eight users. We optimized the recognition procedure for a high precision in order to minimize the risk of false positive detection of arbitrary movements (insertion errors). An average recall of 79% at a precision of 93% demonstrates the intended recognition robustness. This result was achieved as a tradeoff between the number of supported gesture types and the overall recognition performance. In this work we used a set of three gestures only and must assume that additional gestures may deteriorate the overall performance due to confusions. Nevertheless, we confirmed with our approach that even for a practical and complex application as the watch questionnaire, a limited number of gestures is sufficient. In the future we will compare this spotting performance to other approaches such as the Feature Similarity Search proposed in [10], [11].

A limited gesture set also helps to minimize the number of different gesture commands that the user has to learn and remember. In a related study of user ratings we observed that while the focus on performing the gestures decreases, the effort to execute them remains constant [17]. Hence it is essential to restrict the number of gestures to maintain usability even over weeks and months of continuous usage.

We further examined the recognition implementation running on the wearable device. The watch implementation showed a delay of less than 3 ms for feature computation, Viterbi-path processing and subsequent classification at less than 4 KB memory usage. We consider this result as an excellent performance for a resource-constraint device implementation. Furthermore, this is the first time to our knowledge that a continuous spotting performance for gestures had been reported. Consequently, the recognition procedure required only 1% of the total eWatch processing time. We achieved this result by implementing the recognition stack in three stages that subsequently refine the result at decreasing processing rates. These results confirm that there is sufficient remaining processing capacity to include further tasks, compute more features, and spot additional gestures. Our work demonstrates that a gesture-controlled interfaces is feasible for controlling a wearable systems such as a wristwatch.

## References


[1] C. J. Gwaltney, A. L. Shields, and S. Shiffman, "Equivalence of electronic and paper-and-pencil administration of patient-reported outcome measures: A meta-analytic review," *Value in Health*, vol. 11, no. 2, pp. 322–333, March/April 2008.

[2] K. A. Siek, K. H. Connelly, Y. Rogers, P. Rohwer, D. Lambert, and J. L. Welch, "When do we eat? an evaluation of food items input into an electronic food monitoring application," in *PHC 2006: Proceedings of the 1st International Conference on Pervasive Computing Technologies for Healthcare*, E. Aarts, R. Kohno, P. Lukowicz, and J. C. Trainini, Eds. ICST, IEEE digital library, November 2006, pp. 1–10.

[3] C. Narayanaswami, M. Raghunath, N. Kamijoh, and T. Inoue, "What would you do with a hundred mips on your wrist?" IBM Research, Tech. Rep. RC 22057 (98634), January 2001.

[4] Microsoft Corp., "Smart personal object technology," http://www.spotstop.com, last accessed: April 2008.

[5] Van Der Led, "Van Der Led MW1/MW2 phone," http://vanderled.com/onlinestore/product_info.php/ products_id/80, last accessed: April 2008.

[6] MainNav International Corp., "MainNav MW-705 GPS navigation," http://www.mainnav.com/product/product12.htm, last accessed: April 2008.

[7] Tissot, "Tissot T-Touch," http://www.tissot.ch, last accessed: April 2008.

[8] H.-K. Lee and J. H. Kim, "Gesture spotting from continuous hand motion," *Pattern Recognition Letters*, vol. 19, no. 5-6, pp. 513–520, 1998.

[9] J. Deng and H. Tsui, "An HMM-based approach for gesture segmentation and recognition," in *ICPR 2000: Proceedings of the 15th International Conference on Pattern Recognition*, vol. 2, September 2000, pp. 679 – 682.

[10] O. Amft and G. Tröster, "Recognition of dietary activity events using on-body sensors," *Artificial Intelligence in Medicine*, vol. 42, no. 2, pp. 121–136, February 2008.

[11] H. Junker, O. Amft, P. Lukowicz, and G. Tröster, "Gesture spotting with body-worn inertial sensors to detect user activities." *Pattern Recognition*, vol. 41, no. 6, pp. 2010–2024, June 2008.

[12] S. Kallio, J. Kela, P. Korpipää, and J. Mäntyjärvi, "User independent gesture interaction for small handheld devices," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 20, no. 4, pp. 505–524, 2006.

[13] G. Niezen and G. P. Hancke, "Gesture recognition as ubiquitous input for mobile phones," in *DAP 2008: Proceedings of the Workshop on Devices that Alter Perception*, 2008, workshop organized in conjunction with UbiComp 2008.

[14] J. Kim, J. He, K. Lyons, and T. Starner, "The gesture watch: A wireless contact-free gesture based wrist interface," in *ISWC 2007: Proceedings of the 11th IEEE International Symposium on IEEE Wearable Computers*. IEEE, 2007, pp. 15–22.

[15] U. Maurer, A. Rowe, A. Smailagic, and D. Siewiorek, "eWatch: A wearable sensor and notfication platform," in *BSN 2006: Proceedings of the IEEE International Workshop on Wearable and Implantable Body Sensor Networks*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 142–145.

[16] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257 – 286, Feb. 1989.

[17] O. Amft, R. Amstutz, A. Smailagic, D. Siewiorek, and G. Tröster, "Gesture controlleduserinputtocompletequestionnairesonwristwornwatches," in *HCII 2009: Proceedings of the 13th International Conference on Human-Computer Interaction*, ser. Lecture Notes in Computer Science, vol. 5611. Springer, 2009, pp. 131–140, Part 2: Human-Computer Interaction. Novel Interaction Methods and Techniques.