

Accelerometer Based Gestural Control of Browser Applications

Mikko Kauppila, Susanna Pirttikangas, Xiang Su, and Jukka Riekki

Intelligent Systems Group, Infotech Oulu
FIN-90014 University of Oulu

{mikko.kauppila,susanna.pirttikangas,xiang.su,jukka.riekki}@ee.oulu.fi

Abstract. This paper describes our studies in controlling browser applications with an accelerometer based continuous hand gesture recognition system. The system is based on hidden Markov models and includes novel contributions in the areas of tilt compensation (featurization) and gesture segmentation. The efficacy of the contributions is demonstrated, reaching a cross validated (50% fold) recognition rate of about 99% when training data was gathered from a single user. We report the results of a three-stage user study measuring the intuitivity (how well the system fits user expectations), usability and correctness of the system. Our findings suggest that gesturing is a promising alternative interaction mechanism for non-pointing, intermediate paced applications.

Key words: accelerometer, gesture recognition, hidden Markov models, human computer interaction

1 Introduction

The increase of size and resolution, and the decrease in price for displays have led to a situation in which large screens are utilized at train and bus stations, marketing places and other public places, such as libraries, hospitals, universities etc. Usually, the information for the displays comes from different providers through internet. To be able to take full advantage from the large displays, it is convenient to be able to provide different kinds of interaction methods between the users and the displays.

Probably the easiest, cheapest and the most intuitive interaction method for a large display is a touch screen as nowadays, people are used to a mouse-based control which is quite similar to touching the screen. Other possibilities to interact have been suggested as well; data gloves [14], camera-based gesture solutions, such as eye, body and hand tracking [15, 16], and laser pointers [17, 18] to name a few.

The advances in mobile phone sensor technology has created possibilities for ordinary users carrying their own equipment to interact with the displays utilizing gestures. Major vendors have released phones with built-in accelerometers, for example Nokia 5500, Sony Ericsson K850 and Samsung E760. This means

that in the future, more and more people will have ubiquitous access to accelerometers and possibilities to utilize gesture control devices. Wearable sensors attached to the users body as wrist watches or accessories offer interesting possibilities, as well.

In this paper, we study a scenario, where a user approaches a large display, the user is recognized by the display and she can start controlling it with a wearable sensor or a mobile terminal embedded with acceleration signals. The user needs not to be able to touch the screen nor push any buttons, but can start the system right away. The implemented continuous gesture recognition, described in more detail in this paper, provides a natural and an intuitive way to control the browser applications on the large screen.

This paper is organized as follows. In section two, we will outline the previous work in the field and its differences to our present work. In section three, we will describe the communication architecture of the system. In section four, the algorithms used in the recognizer are reported. The setting for our user study and the results are in sections five and six, respectively. Section seven wraps up the paper with a conclusion and suggestions for future work as indicated by the findings.

2 Previous Work

Our recognizer is similar in nature to the hidden Markov model based isolated discontinuous gesture recognizers of [1] and [3]. We note that HMMs are not the only choice, and other models have been proposed in the literature, for example dynamic Bayesian networks [5] and support vector machines [5, 8].

Most of the accelerometer based recognition work (e.g. [1, 3, 5]) rely on user-initiated pre-segmentation of the gestures by having the user hold down a button during gesturing. This is acceptable in some environments, for example when using an accelerometer embedded in a mobile phone, but generally it would be nicer if the system would automatically segment the accelerometer signal. Recently, in the field of machine vision, [8] proposed an algorithm for gesture segmentation based on the observation that gestures have a *ballistic* velocity profile. We adopt a similar mechanism to the accelerometer based recognizer and present a dedicated featurization and distribution model for spotting gestures in a continuous acceleration signal.

The problem of tilt compensation, that is, the reversal of the transformation of the measured linear acceleration caused by orientation of the sensor, has also been approached before, most recently by [3], who proposes a tilt compensation mechanism based on an orthogonal matrix transformation constructed with a Gram-Schmidt process. Our present work extends this idea by having the transformation satisfy certain intuitive geometric constraints.

3 Communication Architecture

In our scenario, the user carrying a 3D-accelerometer approaches a large screen as depicted in Figure 1 a). The user can start applying the gestures to the system right away when the base station (Figure 1 b)) starts receiving data from the sensor.

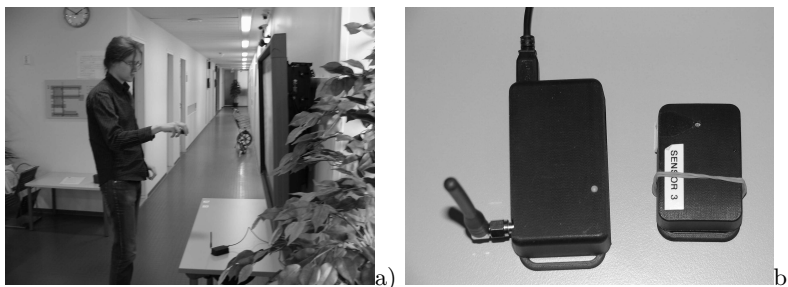


Fig. 1. a) A user holding the sensor device and controlling a browser with gestures. b) The base station and the sensor device in a case.

An overview of the data flow in the architecture is presented in Figure 2. The technical details of each component is presented below.

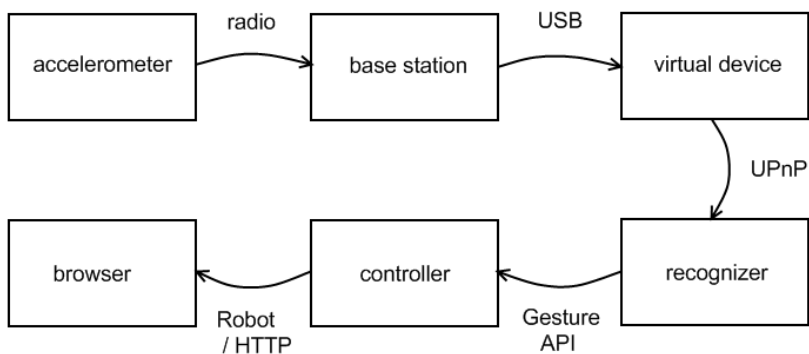


Fig. 2. Data flow

3.1 Accelerometers

At its core, the system relies on acceleration sensors developed at our university. The sensor measures its environment, including acceleration, pressure and temperature, but only the acceleration component is of our interest.

The sensor sends data at 50 Hz over radio link to a base station connected to PC's USB port.

3.2 UPnP Virtual Sensor

To standardize the communication between accelerometers and their client applications, we developed a specification for a UPnP device acting as a *virtual sensor*. The device has a single UPnP state variable that holds a list of the most recent acceleration vectors. The client application then uses a UPnP control point to subscribe to changes in this state variable.

We use threading and buffering to alleviate the performance overhead caused by the UPnP layer (for example, XML and IPC). The first thread accumulates the list of the most recent acceleration vectors, while the second thread sends the list over UPnP at 5 Hz, i.e. with an average size of ten acceleration vectors. We believe the maximum lag (200 ms) caused by the buffering is not detrimental to most applications.

3.3 Gesture API

We use a simple asynchronous Java API based on the observer pattern [9] to decouple the recognizer and its clients. Once a gesture is recognized, the recognizer immediately sends the name of the recognized gesture (for example, "clockwise") to its registered clients. Each client then maps the gesture name to an application-specific action.

3.4 Browser Control

We have implemented two browser controllers. For general browser control, we use a controller based on Java Robot class that works by mapping recognized gestures to keyboard signals that trigger keyboard shortcuts of the browser.









The second controller is designed for the Photo Album application and works by mapping gestures to HTTP requests used to control the Photo Album.

Figure 3 shows the mappings from gestures to their corresponding commands in both controllers.

4 Recognizer

4.1 Segmentation

Similar to [8], we extract ballistic sub-sequences from the continuous input signal. These sub-sequences are then used as input to the HMM-based classifier.

	left	prev. link
	right	next link
	up	scroll up
	down	scroll down
	counter-clockwise	history back
	clockwise	history forward
	punch	click link
	pentagram	refresh





	left	prev. image
	right	next image
	counter-clockwise	fast rewind
	clockwise	fast forward

Fig. 3. Mappings from gestures to commands, general browser controls on the left, Photo Album controls on the right

Let $\mathbf{a}(t) = (a_x(t), a_y(t), a_z(t))$ denote the continuous acceleration signal, and $\mathbf{a}[t] = (a_x[t], a_y[t], a_z[t])$ denote the discretized acceleration signal. Let Δt denote the time step used for discretization (in our system $\Delta t = 20ms$). For segmentation, we use the following feature vector:

$$\mathbf{f}(t) = (|\mathbf{a}(t)|, |\mathbf{a}'(t)|, |\mathbf{v}(t)|)$$

where $\mathbf{a}'(t)$ denotes the derivative of the acceleration and $\mathbf{v}(t)$ denotes the integral of the acceleration (i.e. velocity). The derivative is approximated by central differences at $\mathbf{a}[t - 1]$:

$$\mathbf{a}'[t] = \frac{\mathbf{a}[t] - \mathbf{a}[t - 2]}{2\Delta t}$$

Velocity is approximated with artificial damping to avoid accumulation errors:

$$\begin{aligned} \mathbf{v}(0) &= (0, 0, 0)^T \\ \mathbf{v}'(t) &= \mathbf{a}(t) - c_d \mathbf{v}(t) \end{aligned}$$

Where c_d is the damping constant. We have observed that values near $c_d = 20/s$ produce good results. This differential equation is discretized via the following numerical approximation:

$$\begin{aligned}\mathbf{v}^*[t+1] &= \mathbf{v}[t] + \Delta t \mathbf{a}[t] \\ \mathbf{v}[t+1] &= e^{-c_d \Delta t} \mathbf{v}^*[t+1]\end{aligned}$$

To perform the actual segmentation we use a two-state HMM where the first state corresponds to non-gestural and the second state to gestural segments. The hidden states are recovered using the Viterbi algorithm (e.g. [4]). To model the emission of feature vectors in each state we use a multivariate exponential distribution (assuming statistical independence of features), which seems to fit best into observed feature vector plots.

The emission parameters are estimated from training data, which includes five long non-gestural input sequences and 311 short gestural input sequences.

The transition probability matrix $A = \{a\}_{ij}$ is assumed symmetric for simplicity, that is:

$$A = \begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}$$

The transition probability p is chosen in an ad hoc fashion and acts as a threshold. Values in the lower range (0.001 – 0.1) seem to give good results.

Figure 4 and Figure 5 show examples of segmentation.

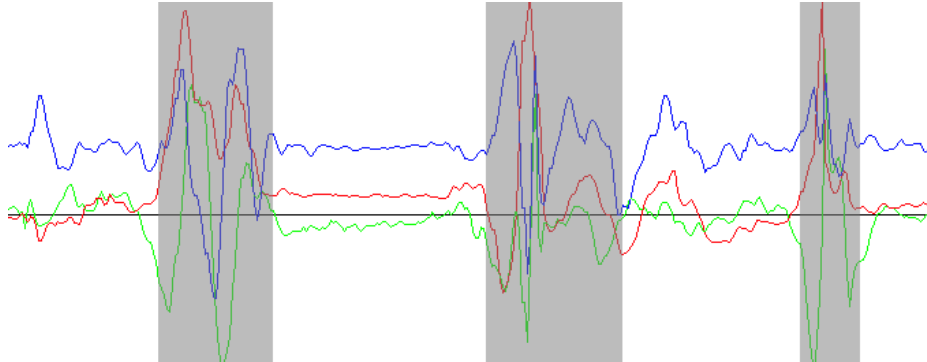


Fig. 4. Segmentation of gestural movements (clockwise, punch and right gestures). Red, green and blue lines denote x, y and z axis, respectively. Segmented gestures are darkened.

4.2 Gesture Featurization

Once the segmentator has extracted a tentative gesture, the classifier still has to decide which class the gesture best fits in. To aid this, the gesture is prepro-

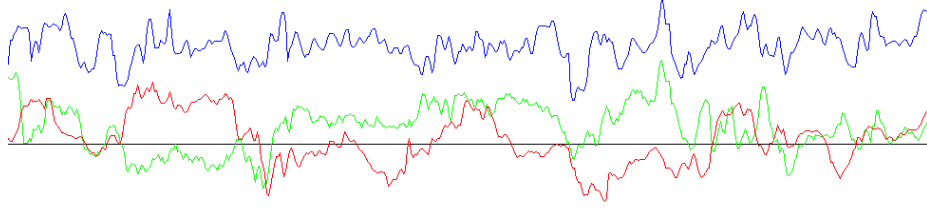


Fig. 5. Non-segmentation of non-gestural movements (moving a coffee mug over a table). Red, green and blue lines denote x, y and z axis, respectively.

cessed (featurized) to reduce unnecessary variability while preserving essential differences relevant to the classification.

Sensor Model Accelerometers not only measure the translational acceleration of the sensor, but also the gravitational effect. More formally, the measured acceleration consists of two components: the dynamic component $\mathbf{a}_d(t)$, and the static component \mathbf{a}_s , the latter of which is time independent and given simply by:

$$\mathbf{a}_s = (0, 0, g)^T$$

Where g is the gravitational constant estimated from sensor data under rest. The final measured acceleration, $\mathbf{a}(t)$ is then given by

$$\mathbf{a}(t) = R(\mathbf{a}_d(t) + \mathbf{a}_s)$$

Where R is an orthogonal matrix describing the orientation of the sensor. We are interested in decoding $\mathbf{a}_d(t)$ from the given measured acceleration $\mathbf{a}(t)$.

Estimating Gravity The static component of the measured acceleration *after* transformation by R , i.e. $R\mathbf{a}_s$, is estimated by assuming it to be constant during the gesture, and assuming that it's the "most explaining" component of the measured acceleration, more precisely

$$R\mathbf{a}_s = \arg \min_{R\mathbf{a}_s} \int_{t_{start}}^{t_{end}} |\mathbf{a}(t) - R\mathbf{a}_s|^2 dt$$

It's easy to see that the solution to this minimization problem is the mean of the measured acceleration sequence.

We can somewhat improve the estimate by taking into the gravitational length constraint $|R\mathbf{a}_s| = g$. The constrained minimization problem can be solved via Lagrange multipliers. The solution is the mean of the measured acceleration sequence rescaled to length g (the direction of the estimated vector is unchanged).

Using the mean to estimate gravity can be problematic since a gestural acceleration sequence is highly variant due to its ballistic nature. Therefore we have also experimented with using a smoothed mean to estimate gravity at each sample point separately. The smoothed mean $\bar{\mathbf{a}}(t)$ is estimated from the acceleration signal $\mathbf{a}(t)$ via the following differential equation:

$$\bar{\mathbf{a}}'(t) = k(\mathbf{a}(t) - \bar{\mathbf{a}}(t))$$

which is discretized as

$$\bar{\mathbf{a}}[t + 1] = \bar{\mathbf{a}}[t] + (1 - e^{-k\Delta t})(\mathbf{a}[t + 1] - \bar{\mathbf{a}}[t])$$

We have noticed that values of k in the range 100 – 200 give good results. We note that this smoothing process must be started before the segmentation is done. We start the process by setting $\bar{\mathbf{a}}[0] = \mathbf{a}[0]$.

Tilt Compensation To recover $\mathbf{a}_d(t)$, we still need to approximate the inverse of the orientation matrix R . Pylvänäinen [3] does this by arbitrarily setting the estimated $R\mathbf{a}_s$ as one of the basis vectors of the orientation matrix R and finding the other basis vectors via a Gram-Schmidt process.

Our basic idea is to generate a rotation matrix R^{-1} around the axis $\mathbf{a}_s \times R\mathbf{a}_s$ such that $R^{-1}(R\mathbf{a}_s) = \mathbf{a}_s$. The formulae for doing this are well known, the following being from [10].

Let $\alpha = \arccos \mathbf{a}_s \cdot (R\mathbf{a}_s)$ denote the angle between the "desired" untransformed static component \mathbf{a}_s and the transformed static component $R\mathbf{a}_s$. Let $\mathbf{v} = \mathbf{a}_s \times R\mathbf{a}_s$ be the axis of rotation, and let $\mathbf{u} = \frac{\mathbf{v}}{|\mathbf{v}|} \equiv (x, y, z)$. Then

$$R^{-1} = \mathbf{u}\mathbf{u}^T + (\cos\alpha)(I - \mathbf{u}\mathbf{u}^T) + (\sin\alpha)S$$

where

$$S = \begin{pmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{pmatrix}$$

and finally

$$\mathbf{a}_d(t) = R^{-1}(\mathbf{a}(t) - R\mathbf{a}_s)$$

The above process fixes the axis of rotation so as to intuitively induce the "straightest" rotation possible. The Gram-Schmidt process has no such guarantees.

Power Normalization Similar to [1], we normalize the power of the acceleration sequence. Instead of min-max-scaling, however, we normalize the data frame matrix formed by the acceleration vectors (with respect to Frobenius norm).

Tempo Normalization Finally, like [1] we rescale the gesture tempo so that all gestures have 30 samples. Downscaling is handled by box filtering, upscaling via linear interpolation.

4.3 Estimation

To teach the system, a single person recorded 16 samples of each gesture class. The person performed the gestures with varying speeds and directions to aid generalization performance. We didn't find any improvements by creating additional samples via simulation (e.g. the white noise of [1]). This could be explained by the featurization, which seeks to eliminate unwanted variation anyway. Also, noisification mostly just adds a covariance prior to the HMM emission parameters.

Each gesture is associated with a 12-state hidden Markov model. Although many researchers (e.g. [1]) have noted that usually a smaller set of states suffices, we found that more complex gestures (e.g. a pentagram) require more expressive models. The emissions of feature vectors in each state is modelled with a single three-dimensional Gaussian.

We use the Bakis, or left-to-right, transition model (e.g. [4]). Initial parameter estimates are obtained via a piecewise linear regression method [6]. Similar to [7], we found that the regression method of [6] should be modified for better estimates.

The piecewise regression algorithm of [6] can be understood as a tree search. We begin with a single segment, and iteratively choose a segment to be split into two sub-segments. For finding the optimal split point, [6] suggests using a maximum heuristic, whereas [7] uses a balancing heuristic. Both [6] and [7] use a breadth-first search strategy. We have observed that the best segmentations (in terms of residual error with a fixed segment count) are obtained via combining the maximum heuristic of [6] with a greedy strategy that always splits the segment with the highest remaining residual error. This strategy-heuristic combination results in less than 27% of the residual error obtained via any other combination.

The initial estimate is then iteratively improved via the Baum-Welch algorithm (e.g. [4]).

4.4 Classification

A gesture is classified by scoring it against each HMM via the forward-backward algorithm (e.g. [4]) and choosing the gesture class corresponding to the HMM with the highest score.

5 User Study

To test the system, we performed a user study with a sample size of eleven subjects. All the testees were very familiar with browsers and computers. Five

of the adult testees were female and six were male. The test was divided into three stages plus a questionnaire, each measuring different aspects of the system.

The subjects were observed during the study. Observers made notes on the performance of the subjects. Unfortunately, no video recording was performed. This constitutes some loss of data due to human errors (inalert observers). The amount of missing data in each measurement, if any, is reported in its corresponding result section.

5.1 Blind Stage

In the first stage, the subjects were asked to freely use the system for general browser control without any prior training or help. This stage measures the intuitiveness of the system: how naturally the users utilize the system, and how well the users are able to discover the associated mappings from gestures to browser actions.

5.2 Task Stage

In the second stage, the subjects were revealed (trained) how the gestures are supposed to be performed, and were provided a list of mappings from gestures to browser actions, similar to the one seen in Figure 3. The users were then asked to solve a specific browsing task, which included finding the name of the spouse of Edgar Allan Poe and the colors of the Sami flag on Wikipedia, starting from a specially crafted page having links to these Wikipedia pages.

The task was designed so that it required using most of the available browsing commands (selecting and following links, scrolling, and moving in history). In total, the task requires performing a minimum of 14 commands. The stage measures how quickly and efficiently users can solve a real task using the gestural interface.

5.3 Photo Album Stage

In the third and final stage, the subjects were introduced to the Photo Album application, which is a browser-based virtual photo album that allows the user to move between images. The users are then suggested to operate the Photo Album freely for a while. The mappings from gestures to Photo Album actions are shown in Figure 3, and an example view of the Photo Album is shown in Figure 6.

In this stage, we measure four variables describing the accuracy of the system: the number of correctly segmented correct classifications, correctly segmented misclassifications, non-segmented gestures (false negatives) and segmented non-gestures (false positives).



Fig. 6. An example view of the Photo Album application

5.4 Subject Feedback

Last, we asked the user for written feedback relating to the experiments. This gives us an opportunity to reflect the observers' findings against the findings of the subjects.

6 Results and Analysis

6.1 Confusion Matrix

To compute the confusion matrix, we used 50% fold cross validation. On each iteration i , half of the samples of each gesture class were used as training data and the remaining half was used for testing. This produces independent and identically distributed unbiased estimates C_i of the confusion matrix C . The estimates are made successively improving in accuracy by averaging:

$$\bar{C}_k = \frac{1}{k} \sum_{i=1}^k C_i$$

The iteration stops when the relative SVD norm between successive averaged estimates is small enough, or more precisely:

$$\frac{|\bar{C}_k - \bar{C}_{k-1}|}{\frac{1}{2}(|\bar{C}_k| + |\bar{C}_{k-1}|)} < \epsilon$$

We used $\epsilon = 0.005$, which resulted in 16 iterations and produced the confusion matrix seen in Table 1 with a total recognition rate of 99%, which is comparable with previous research [1, 3, 5], although we must stress that training data was gathered from a single user only, as explained in Section 4.3. Element c_{ij} of confusion matrix $C = \{c\}_{ij}$ denotes the relative frequency of classifying a gesture of class i to class j .

Table 1. Confusion matrix

	punch	pull	right	left	up	down	clockwise	c.clockwise	pentagram
punch	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
pull	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
right	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
left	0.00	0.00	0.00	0.97	0.00	0.00	0.02	0.00	0.00
up	0.01	0.00	0.02	0.00	0.96	0.01	0.00	0.00	0.00
down	0.00	0.00	0.00	0.00	0.00	0.95	0.00	0.00	0.05
clockwise	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
c.clockwise	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
pentagram	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Effect of Algorithmic Variants To study the effect of different algorithmic variants, we ran the above confusion matrix computation using these different algorithms. For the sake of brevity, only the total recognition rates are presented.

The effect of the gravity constraint (i.e. constraining the length of the estimated gravity to be g) and smoothed gravity estimates (i.e. using a moving average instead of a "batch" average) as presented in Section 4.2 is summarized in Table 2.

Table 2. Effect of algorithmic variants on recognition rate

	constrained	non-constrained
smoothed	98.76%	98.61%
non-smoothed	94.89%	94.59%

Clearly, a smoothed estimate results in an improvement, while the gravity constraint is less important.

Noisification (by a linear signal to noise ratio of three, which was found optimal in [1]) results in recognition rate of 98.35%.

Effect of Folding Rate We also wanted to know how the folding rate affects the recognition rate. For 25% folding (four training samples against twelve test samples) the recognition rate was 97.35%. For 12.5% folding (two training samples against fourteen test samples), the recognition rate was 94.60%.

6.2 User Study

Blind Stage We found that most subjects (nine of eleven) like to naturally hold the sensor with their palms facing downwards. This is a relaxing pose for the wrist, but the sensor is supported by the fingers only. Two people held the sensor with their palms facing upwards, which conversely results in more support for the sensor, but more tension for the wrist.

Generally speaking, the subjects had great difficulty finding the gesture-action associations. Most subjects found zero to two such associations, the most common ones being moving between links (associated with gestures "left" and "right"). We note however, that one person found all except one association. This suggests that once the general mechanism (discrete, translational nature) is discovered, finding rest of the gestures becomes easier.

Almost half of the subjects (five of eleven) had problem discovering the click action associated with the punch gesture since their hand movements were constrained to the two dimensional plane of the wall display ahead of them. This two-dimensional nature of gesturing was already noted by [2], who subsequently constrained their featurization to two dimensions. It's of course possible that the two-dimensional wall display may have acted as a psychological factor.

Another common phenomenon (four of eleven) was performing gestures too slowly (or to use the terminology of [8], *mass-spring* movements). This resulted in false negatives (gestures going unnoticed) and false positives (some relatively fast mass-spring movements still passing the segmentator).

One browser-specific problem was that subjects routinely thought that movement between links is performed by moving their hands up and down, whereas the correct movements would have been left and right. We believe this is explained by the visual layout of links on most web pages.

Task Stage The subjects (ten of eleven, one missing) performed the task in a mean time of 150 seconds, with a standard deviation of 67 seconds (minimum 88 seconds, maximum 276 seconds).

Given that the task required a minimum of 14 commands to be performed, this translates to about 11 seconds per command. We note, however, that almost all subjects had to perform more than 14 commands.

Some specific types of problems with gesturing in general browser control include what we term the *recoiling problem*: for example, a fast hand movement to the right followed by a recoiling movement to the left, constituting a subsequent of recognition of both "right" and "left" gestures where only a "right" gesture was intended. Another problem is a recoiling horizontal hand movement

(left-right or right-left) with a little vertical component (up or down) misclassified as a circular movement (clockwise or counterclockwise). There were also some problems with false positives: for example, a subject quickly pointing to the screen to show something interesting to the observers triggering a false gesture.

Interestingly, two subjects still confused "left" and "right" gestures (associated with moving between links) with "up" and "down" gestures (associated with scrolling), even after the short training period. It seems that "unlearning" the original association takes time which suggests that the new associations really are counterintuitive.

Photo Album Stage The subjects performed a total of 99 commands, constituting an average of 14 commands per subject when the results were gathered from seven subjects (four missing). All users tried to perform all commands (previous image, next image, fast rewind, fast forward).

We note that 94 commands (95%) were correctly segmented and correctly classified. This suggests that the classification results of Section 6.1 generalize fairly well to the user-independent case after a short training period, even though the original training data was gathered in a user-dependent setting as explained in Section 4.3. Four commands (4%) were false positives. The most common source of false positives is the recoiling problem introduced in Section 6.2. There was also a single false negative, where the user performed a movement so slowly that the segmentator didn't catch it.

Subject Feedback First, we asked the subjects five constrained questions.

- *Did the system work as you expected?* Three yes, seven no, one missing.
- *When you made a gesture, did the system perform the desired action?* Average 3.0 (on a scale from one to five).
- *How easy was the sensor to hold?* Average 3.0 (on a scale from one to five).
- *Could you imagine using a wearable sensor constantly?* Seven yes, four no.
- *How concerned are you about the loss of privacy associated with a wearable accelerometer?* Average 2.4 (on a scale from one to five).

Second, we asked the subjects for free-worded feedback about the pros and cons of the interface.

One subject complained about the lack of feedback. Keyboard, mouse and touch based interaction has the advantage of natural tactile feedback. Gesturing is more similar to speech and machine vision in this respect. Generally, gestural feedback has to be implemented into the application (for example, [2] propose a visual feedback method.) Another idea is to use force feedback (e.g. the vibration mechanism in many mobile phones) to confirm the user's actions - this could be easier to incorporate into existing applications.

Four subjects expressed concern over their outward appearance, e.g. "I look stupid" or "Others will see what I'm doing". This is analogous to vision or speech based interaction methods. We believe that the strength of this concern depends

on the application area. Private web surfing is clearly one area where people feel concern towards exposure, whereas, say collaborative user interfaces are a total opposite (one subject expressed the idea of using gestural control for such collaborative environments).

One subject was worried about ergonomic factors - how much stress does long-term use of gestural interfaces cause to hands? Some research has already been done in this area, e.g. [13].

Five subjects complained about the unintuitivity and learning overhead, yet three subjects told that the system was easy to learn. This agrees with our findings: gestural interfaces still require a training period.

Positive aspects reported by the subjects include the novelty, naturality and unintrusiveness of the interface.

7 Conclusion and Future Work

Our gestural interface is inherently *non-pointing* and aimed at issuing discrete commands as opposed to, say, continuous coordinates. The downsides are clearly seen in general browser control, where moving between links by discrete gestures becomes a tedious exercise in hand waving - and as the results point out, these additional gestures used to simulate pointing can be highly unintuitive as well. However, for applications that don't require pointing, such as the Photo Album, a gestural interface seems very promising. One area of future work is to identify or engineer non-pointing applications, preferably with feedback. For example, gesturing could work well in internet imageboards that basically require only scrolling and moving between boards, and possibly refreshing the page to see new images.

False positives still pose a problem. Interpreting every ballistic acceleration subsequence as a gesture simply is not robust enough: some kind of rejection mechanism is needed. Requiring the users to remain relatively steady during non-gesturing restrains wide-scale acceptance of gesture recognition. Our most promising venture for future research in this area lies in the application of garbage models to combine segmentation (spotting) with recognition (classification). Recently, [12] proposed a similar approach in gesture recognition based on whole body motion sequences. We have found that combining standard feature vectors concentrating on the directionality of the acceleration with specifically designed spotting features concentrating on the magnitude of the acceleration (e.g. the ones used in our segmentator) produces especially robust results. The techniques still need more refinement to be able to cope with highly fluent, non-isolated gestures, for example a sequence of numbers, as proposed in [5].

The recoiling problem could be solved by avoiding the use of overly simplistic gestures which are easily performed "by accident", e.g. the simple thrusts to left or right, and rely on deliberately "intentional" geometric shapes instead. For example, [11] proposes a gesture set for controlling PowerPoint™ presentations that doesn't include any simple linear movements.

Another prospect is the generalization of the algorithms to the case of multiple sensors, e.g. two-handed gestures. This would allow larger gesture sets, and probably improve robustness as well since two-handed gestures are even more "intentional" and harder to perform by accident.

Acknowledgments This study was funded by the Finnish Funding Organization for Technology and Innovation and companies. Thanks to Iván Sánchez and Marta Cortés for collaboration in the user study. Thanks to Janne Haverinen for help with the sensor technology. Finally, thanks to all the testees.

References

1. Mäntyjärvi, J., et al: Enabling fast and effortless customisation in accelerometer based gesture interaction, MUM (2004)
2. Kallio, S., et al: Visualization of hand gestures for pervasive computing environments, AVI (2006)
3. Pylvänäinen, T.: Accelerometer Based Gesture Recognition Using Continuous HMMs, Springer-Verlag Berlin Heidelberg (2005)
4. Rabiner, L., Juang, B.-H.: Fundamentals of Speech Recognition, Prentice Hall (1993)
5. Cho, S.-J., et al: Two-stage Recognition of Raw Acceleration Signals for 3-D Gesture-Understanding Cell Phones, (2006)
6. Ge, X., Smyth, P.: Deformable Markov Model Templates for Time-Series Pattern Matching, ACM SIGKDD (2000)
7. Koho, K., et al: Footstep Pattern Matching From Pressure Signals Using Segmental semi-Markov Models, EUSIPCO (2004)
8. Vitaladevuni, S., et al: Ballistic Hand Movements, AMDO (2006)
9. Gamma, E., et al: Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional Computing Series (1995)
10. Neider, J., et al: OpenGL Programming Guide: The Official Guide to Learning OpenGL, Release 1, Addison-Wesley (1993)
11. Lee, H.,K., Kim, J.,H.: An HMM-based Threshold Model Approach for Gesture Recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence (1999)
12. Yang, H.,D., et al: Robust Spotting of Key Gestures from Whole Body Motion Sequence, FGR (2006)
13. Nielsen, M., et al: A Procedure For Developing Intuitive and Ergonomic Gesture Interfaces for Man-Machine Interaction, Technical Report (2003)
14. Eisenstein, J., et al: Device Independence and Extensibility in Gesture Recognition, IEEE Virtual Reality 2003
15. Skaburskis, A, et al: Auramirror: Reflections on attention, ACM Symposium on Eye Tracking Research and Applications (2004)
16. Nickel, K. Stiefelhagen, R.: Pointing Gesture Recognition Based on 3D-tracking of face, hands and head orientation, International Conference on Multimodal Interfaces (2003)
17. Oh, J.-Y., Stuerzlinger, W.: Laser Pointers as collaborative Pointing Devices, Graphics Interface (2002)
18. Peck, C.: Useful Parameters for the Design of Laser Pointer Interaction Techniques, Extended Abstracts of the ACM CHI Conference (2001)