Pers Ubiquit Comput
DOI 10.1007/s00779-011-0451-8

ORIGINAL ARTICLE

A study on automatic recognition of object use exploiting motion correlation of wireless sensors

Stephan Bosch · Raluca Marin-Perianu · Paul Havinga · Arie Horst · Mihai Marin-Perianu · Andrei Vasilescu

Received: 7 March 2011 / Accepted: 19 August 2011

Abstract An essential component in the ubiquitous computing vision is the ability of detecting with which objects the user is interacting during his or her activities. We explore in this paper a solution to this problem based on wireless motion and orientation sensors (accelerometer and compass) worn by the user and attached to objects. We evaluate the performance in realistic conditions, characterized by limited hardware resources, measurement noise due to motion artifacts and unreliable wireless communication. We describe the complete solution, from the theoretical design, going through simulation and tuning, to the full implementation and testing on wireless sensor nodes. The implementation on sensor nodes is lightweight, with low communication bandwidth and processing needs. Compared to existing work, our approach achieves better performance (higher detection accuracy and faster response times), while being much more computationally efficient. The potential of the concept is further illustrated by means of an interactive multi-user game. We also provide a thorough discussion of the advantages, limitations and trade-offs of the proposed solution.

This work is an extension of earlier work presented at the fourteenth annual IEEE International Symposium on Wearable Computers (ISWC 2010) [5].

S. Bosch (✉) · R. Marin-Perianu · P. Havinga · A. Horst
University of Twente, Enschede, The Netherlands
e-mail: s.bosch@utwente.nl

M. Marin-Perianu
Inertia Technology B.V., Enschede, The Netherlands
e-mail: mihai@inertia-technology.com

A. Vasilescu
PROSYS PC SRL, Bucharest, Romania
e-mail: andrei.vasilescu@prosyspc.ro

1 Introduction

Ubiquitous computing imagines an instrumented environment surrounding the user, capable to recognize, interpret and react to the user's activities. An essential component in this vision is the ability of detecting which objects are being used at any moment, in a non-intrusive manner. A number of solutions have already been proposed in the literature: RFID [4, 21, 22, 31], contact switches [26, 27] and power consumption monitoring of electrical appliances in the home [3]. Each of these has its own limitations. RFID systems may erroneously mark an object as being held by the user just because it is in close proximity or, conversely, the user's interaction may be missed when the object is grabbed at a great distance from the RFID tag [31]. The techniques using contact switches and monitoring the power consumption of appliances in the home provide no information on the user's identity and therefore those only provide a suitable solution when the identity of the user either is known implicitly or not important. Furthermore, the use of contact switches requires that there is an actual contact involved in the user action, for example when the object has a knob. Monitoring an appliance's power consumption gives solely an indirect estimate of the usage and is restricted to electrical appliances.

More recently, benefiting from the rapid progress in MEMS manufacturing, inertial and magnetic motion sensors became an alternative technology to be considered. If worn by the user and attached to objects, such sensors can detect that an object is being used by a particular user by correlating their movements. This approach has two important advantages over the previous alternatives: (1) it gives a direct measure based on the actual object use (and not based on proximity to the object, for example) and (2) it provides much more and finer-grained sensor

Published online: 09 September 2011

information, thus making possible to also detect how the user is interacting with the objects (which can lead, for example, to inferring the user's activities). The main limitation of this approach is that it assumes some dynamics involved in the interaction, i.e., the user is supposed to actually handle and move the objects.

This study explores therefore a solution for automatic detection of object use based on wireless sensor nodes outfitted with three-dimensional accelerometer and compass sensors. We equip the objects of interest and the user's arm with sensor nodes that correlate their relevant motion features. The feature extraction, communication and correlation are performed online and cooperatively by the sensor nodes, which guarantees a fast response time to the user. Through detailed simulations and practical experiments, we seek to answer the following questions:

1. What are the signal features that express well and compact the motion information, while remaining computationally simple enough to be implemented on resource-constrained hardware?
2. How can the correlation be done efficiently among the sensor nodes?
3. What are the relevant parameters and trade-offs? How to choose the optimal values?
4. How does our solution compare to existing work?
5. How do we implement the overall system on sensor nodes?
6. What is the performance in practice, and how does it compare to simulations?
7. What are the main problems, limitations and ideas for further improvement?

The remainder of this paper is organized as follows. Section 2 surveys previous work on detecting object use. Section 3 provides the solution overview, covering aspects related to feature extraction, correlation, communication and synchronization. The system performance and trade-offs are analyzed through simulation in Sect. 4. In Sect. 5, our algorithms are compared with existing work in terms of performance and computational effort. Next, Sects. 6–8 successively present all the practical details, from implementation on sensor nodes to practical experiments with user activities and an interactive game. Finally, Sect. 9 discusses the results and formulates the conclusions.

2 Related work

Performing or enhancing activity recognition using information on what objects the user is currently manipulating is already a well-established concept [21, 22]. The actual detection of a user's interaction with a particular object is one of the key problems that needs to be solved.

For situations in which only the interaction itself is interesting or in which the identity of the user is known implicitly, e.g., in a single-user environment, the detection of a user's interaction with objects can be implemented in a relatively straightforward manner by adding switches or other simple sensors to the points of interaction, such as doors, knobs, handles and levers [26, 27]. A more complex method is employed by Bauer et al. [3], who infer the use of kitchen appliances by analyzing the electrical current on the power line.

However, in the general sense, e.g., for a multi-user environment, it is often necessary to find out *who* is using a particular object by establishing an association between the object and the user. Therefore, it is not sufficient to use a simple switch or contact sensor. Moreover, the use of an infrastructure becomes less practical, as this would tie the object to an instrumented environment.

A common solution is to detect whether a particular user, or rather his arm, is in close proximity to an object by employing the RFID [4, 8, 31] technology or other RF-based solutions [6]. Each object is equipped with an RFID tag, while the user wears a reader on his hand or lower arm. Once the reader is in close proximity to the object, the system assumes that the object is being used. Important advantages of RFID tags are that they are cheap and small and that no batteries are required. Unfortunately, spatial proximity information alone is often not enough to reliably detect the use of an object by a particular user, mainly because other nearby objects can also be detected [31], e.g., when the user moves in their proximity [10]. This problem is most prevalent when the communication range is large relative to the density of objects in the environment. However, limiting the communication range to alleviate this problem can affect performance. Therefore, a trade-off between range and reliability needs to be found [8]. Another problem is that RFID readers need a relatively large antenna to attain sufficient range while limiting power consumption, which makes the design of a suitable bracelet or glove difficult according to Berlin et al. [4]. The work by Berlin et al. compares several RFID systems used in context recognition research and, even with their own well-tuned antenna, the range is still limited to about 14 cm. This may be problematic when the object is large and handled at an unexpected position [8].

A problem that cannot be solved with the RFID technology is the detection of *how* the object is being manipulated by the user [30]. For example, with a claw hammer, one can drive nails into a wooden board but also pull them out again. This can be distinguished using sensors on the user, but the object itself can also provide vital information about the activity it is involved in. To exploit this, objects can be equipped with sensing, processing and communication capabilities. Such objects are called smart objects

[16], sentient artifacts [9, 15], smart artifacts [13] or cooperative artifacts [25].

We use this technology for the detection of object handling by exploiting the fact that typically the user is holding the object in his hand and the performed activity involves at least some movement. For such scenarios, we investigate the detection of the simultaneous motion of the object and the user's arm. We use hardware that is directly suitable to build a smart object, allowing it to not only detect that it is being used by someone, but also in what manner. We attach to objects and users wireless sensor nodes equipped with motion sensors. By matching the movement of the object and the movement of the user's arm, we can assess whether the object is being used by that user, thereby establishing an *implicit connection* based on the *context proximity* [13] between the user and the object.

The concept of associating two entities based on their common movement has been explored in a couple of application scenarios. For example, for a transport and logistics application, Marin-Perianu et al. [19] describe a method to determine whether wireless sensor nodes attached to transportation items are moving together based on raw accelerometer data. Our application, however, has different requirements and challenges, because it involves human motion. The characteristics of human motion are different compared to motion of transportation items, mainly because the sensors can rotate freely in any direction.

Aylward et al. describe Sensemble [1], a system of compact, wireless sensor nodes meant to capture expressive motion when worn at the wrists and ankles of a dancer. The research uses inertial sensors, i.e., both accelerometers and gyroscopes, and time-domain covariance calculation to obtain a measure for the movement correlation between different sensor modules. The raw sensor data of the Sensemble nodes are streamed to a central processing unit through a high-bandwidth radio. Unfortunately, the details of the performed processing are omitted. Also, the use of infrastructure is less desirable and practical for our application, as this would tie the objects to an instrumented environment. We aim to implement the correlation algorithm on the nodes themselves with only limited communication needs.

The work by Lester et al. [17] uses motion sensors to determine whether two objects are carried by the same person by exploiting the periodic nature of human walking. This makes correlation in the frequency domain possible, reducing the effect of communication latencies and avoiding the need for precise synchronization. Also using frequency-domain analysis, Mayrhofer et al. [20] exploit shared movement patterns to authenticate communication between wireless devices. The user can pair the devices for secure communication simply by shaking them together. However, such frequency-domain methods are less likely to perform well with generic non-repetitive movement, such as for a human arm manipulating an object.

The idea of establishing a connection between devices by—for example—shaking them together was coined earlier in the Smart-Its project [13]. Although the details of the employed algorithm and parameters are not described, the authors briefly mention the use of time-domain correlation of accelerometer data to decide whether objects are moving together [23].

Hinckley et al. [12] also use motion sensing to establish device association, in this case between tablet computers. By bumping the devices against each other, a relation is established. Unlike the other time-domain solutions discussed above, this work does not use correlation of the full sensor signals. Rather, the individual devices detect 'bump' events in their accelerometer's signal and wirelessly try to match the time-wise occurrence of these among each other. Human activity involving object use does not necessarily involve bumps or other sharp impact events, making this less suitable as a generic solution to our problem. However, the detection and matching of such impact events could be employed to improve the reliability of our solution.

Buettner et al. [7] use movement data to determine that objects are used and how these are used. The movement data are collected by means of RFID technology, but unlike the normal RFID tags, these so-called Wireless Identification and Sensing Platforms (WISPs) [24] include processing and sensing capabilities, such as an accelerometer. This alleviates the battery problems incurred in normal wireless sensor network designs. However, the described solution omits determining *who* is using the objects in the room; the RFID readers are mounted in the ceiling and not on the users. When using an RFID bracelet with inertial sensors [4, 8], this WISP technology could be used to combine the advantages of RFID and proximity-based solutions with our movement-based approach.

The work by Fujinami et al. [10, 11] explores using the correlation of raw accelerometer signals for object–user association. Much like our solution, Fujinami et al. use the statistical correlation coefficient to establish an association between an object and a particular user. However, our experiments indicate that significant rotation can make the accelerometer-only solution less accurate, mostly through the influence of gravity on the accelerometer. Therefore, we base our solution on the fusion between the accelerometer and compass sensor data, thereby also involving rotation information. Additionally, we use correlation of motion features instead of raw sensor data, thus reducing significantly the overall processing and communication requirements. In Sect. 5, we compare two of the algorithms by Fujinami et al. with our own.

Table 1 summarizes the related work along the following lines: the technological solution, the method for

Table 1 Overview of related work

Solution	Method (direct/indirect)	Main limitations/ problems	Multi-user, multi-object environments	Feasible on constrained hardware	Use-mode detection possible
Simple contact switches [26, 27]	Contact (direct)	Limited to binary actions	No multi-user	Yes	Limited
Electrical appliances [3]	Current consumption (indirect)	Limited to electrical appliances	No multi-user	Yes	Limited
RFID [4, 8, 31]	Spatial proximity (indirect)	False positives/range problems	Yes	Tags—yes, readers—no	Limited
Motion: time-domain correlation of raw data [1, 10, 11, 19, 23]	Correlated motion (direct)	Rotation for accelerometer-only solutions	Limited due to computational effort	Yes	Yes
Motion: frequency correlation [17, 20]	Similar motion frequency (direct)	Periodic motion needed	Very limited due to computational effort	No	Yes
Motion: impact correlation [12]	Impact/bumping (direct)	Impacts with/of objects needed	Yes, with significant loss of accuracy	Yes	No
Motion: feature correlation [5]	Correlation motion (direct)	At least some motion needed	Yes	Yes	Yes

detecting the object use and whether it gives a direct or indirect estimate of the actual use, the main limitations or issues, the ability of distinguishing multi-user, multi-object cases, the feasibility of implementation on resource-constrained hardware and the possibility of inferring how the objects are used. Our proposed solution (motion feature correlation) is also represented in this overview table for comparison. The main benefits are that with less computational effort (due to the feature extraction), we can achieve high detection accuracy and fast response times. Additionally, our solution can facilitate complex activity recognition by providing information on how the object is being used.

3 Solution overview

Our solution is based on smart objects (also called smart artifacts, sentient artifacts or cooperative artifacts [13, 15, 25]), which envelop sensing, processing and communication capabilities. Each object is equipped with a wireless sensor node with three-dimensional accelerometer and compass sensors. For a pair of sensor nodes under consideration, movement measurements are correlated in the time domain using the Pearson product-moment correlation coefficient. The accelerometer is used to measure linear motion, whereas the compass sensor is used to measure rotary motion. A node performs the correlation calculation using its local measurements and those communicated wirelessly from the peer node. To reduce communication and processing efforts and to improve the correlation performance, the raw sensor signals are first processed into concise feature values before being communicated and used in the correlation. Unlike actual activity recognition, temporal segmentation of the sensor data is not necessary to perform the correlation. For this application, we are not so much interested in the composition of the movement, but rather in the correlation between the movements of the sensors.

In the following, we present our solution in detail. We first describe how we avoid relative orientation dependencies in the motion correlation process and why a compass sensor is employed. Thereafter, we describe the used signal features and how these are extracted from the sensor signals. Subsequently, we outline the synchronization of feature extraction, the communication details, how our solution deals with the lack of movement and finally the correlation algorithm itself. We finish this section by summarizing the parameters involved and the associated trade-offs.

3.1 Orientation dependencies

An important problem that needs to be solved is that the orientation of the sensors is likely to differ, which means that the individual axis signals of the 3-D sensors cannot be correlated directly. Therefore, the sensor signals need to be preprocessed to remove the orientation dependency. Only then, the correlation coefficient will produce reliable and accurate results for sensors with unknown relative orientation.

One of the solutions proposed by Fujinami et al. [10] to solve this problem is by taking the magnitude of the raw accelerometer signal. This way, only the intensity of the

motion is fed to the correlation algorithm and the relative orientation of the sensors should have no influence. However, rotary motion can significantly influence the performance of this design, because the accelerometer measures the vector sum of acceleration and gravity (also known as *specific force*). Figure 1a schematically shows an extreme example of the problem. Two accelerometers are rigidly mounted on the opposite ends of a movable beam. Clearly, the movement of the two sensors is always physically correlated. The solution using correlation of the acceleration magnitudes works fine when both sensors have the same approximate movement direction relative to gravity, i.e., the beam moves such that both are moving up or both are moving down. However, when the beam rotates such that one sensor moves up and the other sensor moves down, i.e., with the pivot point between them, the problem emerges: one sensor experiences an acceleration **a** pointing in the general direction of gravity **g**, while the other experiences an acceleration that has a direction opposite to **g**. When **a** is small relative to **g**, the magnitude of the measured acceleration $|\mathbf{a_m}|$ will in that case rise above $|\mathbf{g}|$ for one sensor and drop below $|\mathbf{g}|$ for the other. Only if the acceleration a caused by the rotation is large enough, the acceleration magnitude $|\mathbf{a_m}|$ will rise above $|\mathbf{g}|$ for both sensors.

Rotary motion can therefore severely impede the performance of accelerometer correlation. Figure 2 shows the accelerometer magnitudes from the experiment described above. The beam is rotated back and forth with around 90 degrees. As expected, the signals look anti-correlated. Because magnitude signals (movement intensities) are compared, anti-correlation has no specific meaning due to the lack of directional information and is therefore interpreted as no correlation at all. Overall, the correlation of the signals in Fig. 2 is very low.

In practice, this situation can arise quite easily, for instance, when the user is rotating his wrist as shown in Fig. 1(b): The pivot point is the user's wrist, the bracelet is on top of the user's wrist and the object is in the user's hand and thereby below the pivot point. Rotating the arm as indicated in the figure triggers the issue. As a partial workaround, we coarsely compensate for gravity, as explained in Sect. 3.2. Because the compensation is not perfect, this does not solve the gravity problem completely, and significant rotation can still reduce performance.

When the sensor is subjected to much rotary motion, the accelerometer correlation becomes less reliable. Therefore, we need (1) a means to detect this situation and (2) a sensor modality that provides a means to reliably correlate rotary motion. As a solution, we add a compass sensor, which measures its three-dimensional orientation relative to the Earth's magnetic field. Much like the accelerometer, we process the compass signal into a series of feature values that correspond to the motion intensity in an effort to lose its orientation dependency. To achieve this, we determine the angle between successive compass orientation measurements, thereby obtaining a measure for the angular speed. This feature value is useful both as a means to detect situations with significant rotary motion and as means to correlate rotary motion between sensors. To solve the accelerometer issue, we combine the accelerometer and compass correlations into a single assessment value, in which the accelerometer correlation gets less weight when there is much rotary motion. This is explained further in Sect. 3.4.

3.2 Feature extraction

The feature values are extracted from the raw signal at regular non-overlapping intervals called *windows*. A new

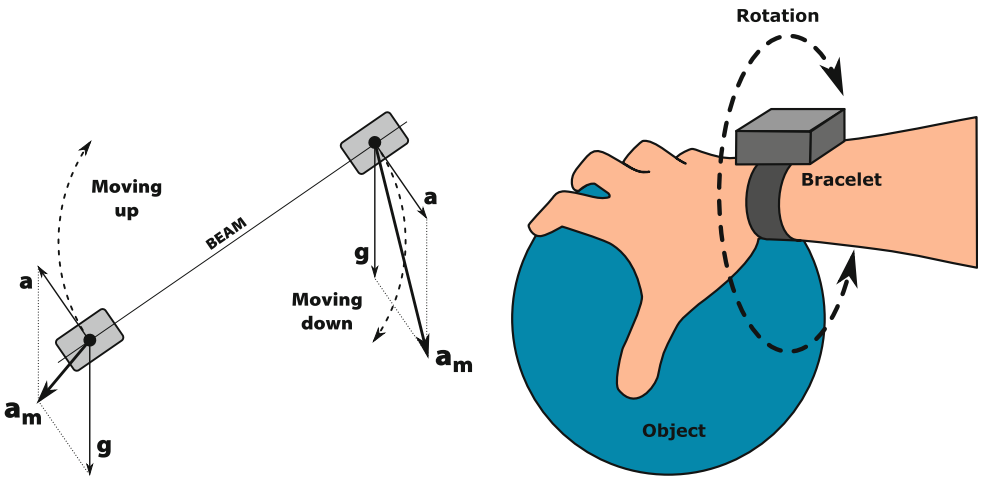

Fig. 1 Gravity influence on accelerometer correlation performance

(a) Schematic overview **(b)** Example arm rotation

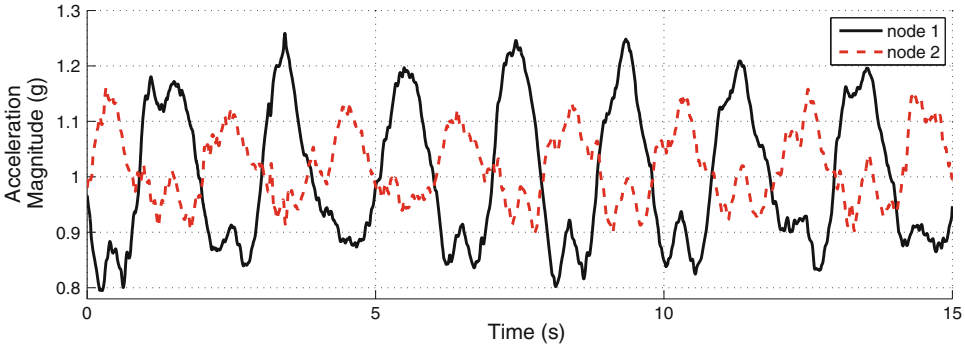

Fig. 2 Example of gravity and rotation affecting accelerometer correlation

feature is computed at the end of such an interval, which means that the length of the windows, i.e., the *window size* W, determines the rate at which features are generated with a given sample frequency f_s, i.e., the *feature frequency* ($f_f = f_s/W$). The window size thereby also determines how many sensor samples are combined into a single feature value.

For our application, the extracted features need to meet the following requirements:

- *Good feature quality* The extracted features must adequately retain the overall motion characteristics, such that the correlation algorithm can reliably assess both the presence and absence of correlated motion.
- *Low processing requirements* The resource limits of sensor node hardware dictate that the processing requirements need to be as low as possible. This means that simplicity is key and that simple features that can be computed with little effort have preference.
- *Low feature size and rate* The extracted features must be small and produced at a low rate to keep bandwidth and processing requirements low. However, a certain minimum level is needed to achieve adequate performance.
- *No orientation dependencies* The features must be computed such that the absolute orientation of the sensor has little or no influence on the produced feature result, as explained in Sect. 3.1.

To meet these requirements, we define a *feature vector* composed of two features that describe the intensity of linear and rotary motion. They are orientation-independent and relatively easy to calculate from the sensor data:

– *Compass rotation angle:* We infer the intensity of rotation during a given time interval by calculating the angle between vectors measured at the beginning and the end of a feature window through the dot product. The *compass rotation angle* f_{cra} feature is calculated from the compass measurements $\mathbf{m}(t) = \langle m_x(t), m_y(t), m_z(t)\rangle$ in the window interval $t = 1 \ldots W$ as follows:

$$f_{cra} = \frac{\mathbf{m}(1) \cdot \mathbf{m}(W)}{|\mathbf{m}(1)||\mathbf{m}(W)|} \tag{1}$$

The feature value is normalized to yield a cosine angle value in the interval $[-1; 1]$. When there is no rotation, $f_{cra} = 1$. The compass sensor needs to be calibrated for offset and scale differences between the sensor's own axes.

– *Mean acceleration magnitude:* To make the accelerometer data insensitive to the current orientation of the sensor, two important steps are taken within a window interval $t = 1 \ldots W$:

1. The raw accelerometer signal $\mathbf{a_r}(t) = \langle a_{r,x}(t), a_{r,y}(t), a_{r,z}(t)\rangle$ is stripped from its offset by subtracting the mean value in the window interval. This coarsely compensates for the gravity component, which depends on the sensor orientation and usually changes slower than the actual acceleration the sensor is subjected to through human motion. Additionally, this step compensates for the effect of offset miscalibration, avoiding the need to calibrate the individual accelerometers.
2. The sum of the absolute vector components is calculated from the three axis components. This has a similar response to the more computation-intensive acceleration magnitude. This discards the vector's direction and retains only its length, resulting in one value per sample that describes the desired intensity.

Summarizing, this *mean acceleration magnitude* f_{mam} feature is calculated from the raw acceleration samples $\mathbf{a_r}(t)$ in the window interval $t = 1 \ldots W$ as follows:

$$\begin{aligned} a_x(t) &= a_{r,x}(t) - \overline{a_{r,x}} \\ a_y(t) &= a_{r,y}(t) - \overline{a_{r,y}} \\ a_z(t) &= a_{r,z}(t) - \overline{a_{r,z}} \\ f_{mam} &= \frac{1}{W}\sum_{t=1}^{W} |a_x(t)| + |a_y(t)| + |a_z(t)| \end{aligned} \tag{2}$$

3.3 Communication and synchronization

To assess the motion correlation, at least one of the nodes needs to have the feature values of both sides available. Therefore, these values are continuously communicated wirelessly to the peer at the instant they become available in a *feature message*. When a feature message is lost on the wireless channel, the corresponding feature on the receiving end is also discarded so that the correlation result is not affected. As long as no new messages arrive, the correlation result is not updated meaning that the detection state is retained.

For a good correlation performance, it is important that the involved sensor nodes sample and generate the features at approximately the same time. If there is too much time skew between the nodes, no correlation is detected even though correlated motion may exist. However, the synchronization demands are not high: If the skew is small enough relative to the feature window size, the performance is not much affected, since the features are calculated from roughly the same time span. Although better synchronization improves the performance, it requires more processing and communication resources. Therefore, we aim for only coarse sampling synchronization between the nodes in real usage.

3.4 Correlation algorithm

The correlation of the two motion features in the feature vector between two nodes is done in the time domain using the Pearson product-moment correlation coefficient:

$$\rho_{A,B} = \frac{\mathrm{cov}(\mathrm{A},\mathrm{B})}{\sigma_A \sigma_B} \tag{3}$$

We use the following formula (the *sample correlation coefficient* $corr_s$) to approximate $\rho_{A,B}$ for the last H values from sources A and B:

$$\begin{aligned} \rho_{A,B} \approx \mathrm{corr}_s(A,B,H) &= \frac{\sum_{n=1}^{H}(A_n - \overline{A})(B_n - \overline{B})}{\sqrt{\sum_{n=1}^{H}(A_n - \overline{A})^2 \sum_{n=1}^{H}(B_n - \overline{B})^2}} \\ \overline{A} &= \frac{1}{H}\sum_{n=1}^{H} A_n \\ \overline{B} &= \frac{1}{H}\sum_{n=1}^{H} B_n \end{aligned} \tag{4}$$

In our case, the correlation is calculated over the last H (correlation history length) features produced at a pair of nodes. The result $\rho_{A,B}$ lies in the range $[-1; 1]$, for which the following situations are distinguished:

- $\rho_{A,B} = 1$: The signals are fully correlated.
- $\rho_{A,B} = 0$: The signals are not correlated.
- $\rho_{A,B} = -1$: The signals are fully anti-correlated.

It should be noted that since both features are vector *magnitudes*, which have no direction, anti-correlation is not interpreted as correlation. Therefore, in our case, correlation values ≤ 0 mean that the movements are not correlated at all.

Using Eq. 4, separate correlation values are calculated for the two feature values f_{cra} and f_{mam}. These results have to be combined into a single value that indicates how well the motion of the two nodes correlates. As explained in Sect. 3.1, the reliability of the accelerometer correlation is sensitive to rotational motion. Therefore, we involve the current compass rotation (f_{cra}) features from both sensors to produce a weighted average of the accelerometer correlation ρ_{mam} and the compass correlation ρ_{cra}. This is done using the following heuristic formula:

$$\begin{aligned} \alpha &= \frac{1}{4} + \frac{1}{8}(f_{\mathrm{cra},1} + f_{\mathrm{cra},2}) \\ \rho_{\mathrm{combined}} &= \alpha\rho_{\mathrm{mam}} + (1-\alpha)\rho_{\mathrm{cra}} \end{aligned} \tag{5}$$

The combined correlation result ρ is the average of both correlations when there is no instantaneous rotation ($f_{\mathrm{cra}}, 1, f_{\mathrm{cra}}, 2 = 1$), and it is the compass correlation alone when both f_{cra} features are at their extreme value ($f_{\mathrm{cra}}, 1, f_{\mathrm{cra}}, 2 = -1$).

The correlation value produced by our algorithm lies in the range $[-1; 1]$. To obtain a discrete decision on whether an object is being held and used by the user, we need to define the thresholds for when the detector status changes from *not used* to *used* and vice versa. These thresholds are not necessarily equal in both directions, yielding hysteresis between the two states.

3.5 Motion detection

If one sensor node is stationary while another sensor node is moving, these nodes obviously cannot be moving together. Communicating feature values and calculating the correlation coefficient is then a waste of resources. Moreover, when sensors are stationary or barely moving, the correlation coefficient becomes sensitive to noise and vibration in the motion signals, and thus less reliable. It is therefore more efficient and reliable to first compare the variance of the movement signals; the correlation coefficient is only calculated when both sensor nodes are actually moving.

It should be noted that because correlation calculation only starts at the moment both nodes are moving, there is a setup phase in which less than the correlation history (H) feature values are available for correlation. This, in effect, reduces the response time of the algorithm when an object is for instance first picked from a table. This may also cause a brief false correlation.

3.6 Algorithm parameters and trade-offs

The operation of the correlation algorithm depends on a set of parameters that directly influence its performance:

- *Sensor sampling rate* (f_s in Hz): A minimum sample frequency is necessary to capture movements with sufficient temporal resolution to prevent aliasing effects and for the correlation to work. Nevertheless, an unnecessary high sample frequency wastes resources on sampling and data processing.
- *Feature frequency* (f_f in Hz): If more features are produced per unit of time, more detail is retained in the data. Also, the response time of the algorithm may improve as changes in the sensor data lead to changes in the feature data more quickly. However, a higher feature frequency increases the communication bandwidth and the processing cost of movement correlation.
- *Correlation history length* (*H in features*): The correlation history length determines the correlation time interval $\left(T_H = \frac{H}{f_f} \text{ in seconds}\right)$ in seconds). On the one hand, a longer interval results in a more reliable and stable correlation, i.e., less sensitive to brief coincidental correlation. On the other hand, a longer interval significantly increases the response time of the algorithm and the processing requirements.
- *Decision thresholds:* The decision thresholds determine when the detector state makes a transition from correlating to non-correlating and vice versa. These thresholds directly affect the accuracy of the assessment. If not chosen carefully, the reliability is decreased with frequent erroneous output. The thresholds also affect the response time of the detector, since the correlation output value exhibits a non-instantaneous (sloping) response.

4 Simulation

To evaluate the trade-offs that exist among performance metrics such as accuracy, response time and resource usage, we perform an off-line evaluation using MatLab before implementing our solution on the actual hardware. The simulation uses raw data from the actual sensors and allows us to freely adjust the algorithm parameters, thus automating the analysis of the trade-offs. We perform numerous experiments with users handling objects equipped with sensors. Using a fast custom TDMA protocol, we collect all raw data at 100 Hz with a synchronization precision better than 10 μs.

4.1 Performance evaluation

In order to analyze the trade-offs, we vary each algorithm parameter individually while each of the other parameters is held constant at a sensible value that will not negatively influence the results. Each constant parameter is chosen such that the effect of the varied parameter is expected to be most apparent. This is not necessarily equal to the optimum value found in an earlier simulation. For example, to make sure that the sample frequency f_s is always high enough to envelop the bandwidth of human motion (mostly below 20 Hz), the feature frequency f_f and correlation history H are evaluated at a relatively high sample frequency of 40 Hz, even though simulations show that it does not necessarily need to be that high for adequate correlation performance.

We explore sampling frequencies ranging from 1 to 100 Hz, feature frequencies ranging from 0.5 to 20 Hz and correlation history sizes ranging from 0.5 to 20 s. The performance is measured in terms of the detection accuracy and response time:

- *Accuracy* is assessed by determining the mean and variance of the correlation value produced for correlated and uncorrelated motion. The mean must lie close to the optimum value, which is 1 for correlated movement and 0 for uncorrelated movement, and the variance should be ideally close to zero.
- *Response time* is the time the algorithm needs to detect a change in the interaction state, i.e., the onset or the end of movement correlation. We assess the response time for the onset and the end of the correlation separately.

4.2 Experiments

The purpose of the first set of experiments is to evaluate the accuracy of the algorithm. In these experiments, three sensors are placed on the user's arm, so that there is continuous correlation among them. Each experiment lasts for one minute. Four different types of movement are considered: lifting a dumbbell weight, moving a ball up and down above the shoulder, making a rowing motion with a wooden stick and making random movements. For each type of movement, five separate experiments are performed. To evaluate the accuracy for correlated movement, we run our algorithm on data from different nodes in the same experiment. To evaluate the accuracy for uncorrelated movement, the data from different experiments are cross-matched.

The second set of experiments assesses the response time of the algorithm. In these experiments, one sensor node is attached to the user's arm and two nodes are attached to objects handled by the user. The user

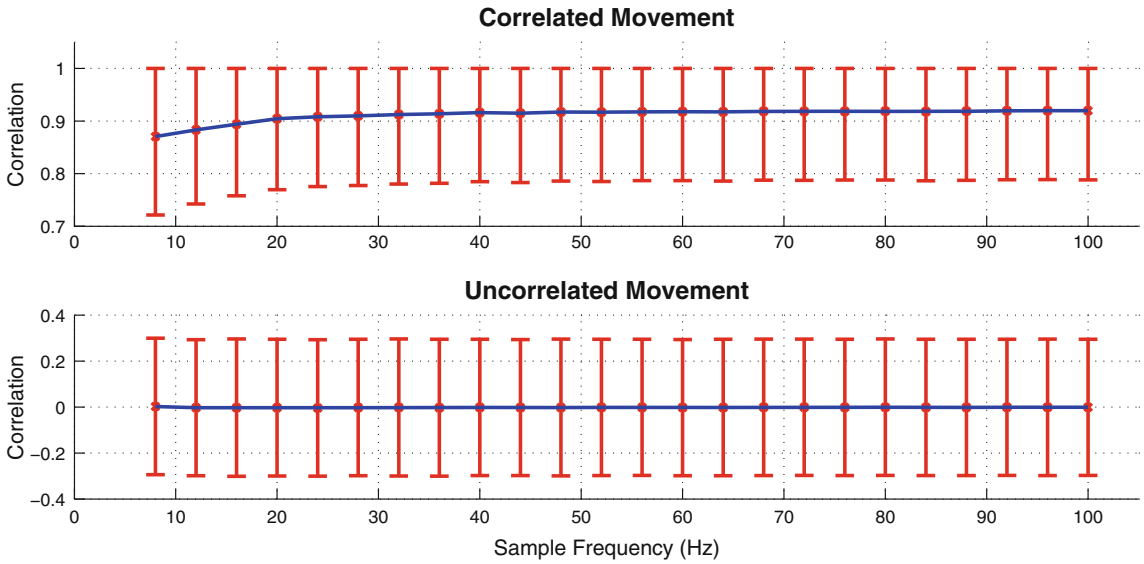

Fig. 3 Correlation output statistics at varying sample frequency

successively interacts with one of the two objects. The objects are equipped with a push button that is pressed by the user when he is holding the object. This method establishes the ground truth for the object usage by interpreting the state of the buttons. Each experiment lasts two and a half minutes. We perform five experiments in which the objects are handled solely by the user and five other experiments in which the objects are constantly kept moving by a second person when the user is not interacting with them.

4.3 Results

4.3.1 Impact of sampling rate

The simulation results presented in Fig. 3 show that increasing the sampling rate up to about 24 Hz has a significant positive influence on the accuracy of detecting correlated motion ($f_f = 4$ Hz and $H = 2$ s). Beyond 24 Hz, the impact is insignificant as shown in the top plot. The bottom plot shows that the sample frequency has no visible influence on the accuracy of detecting uncorrelated motion. Furthermore, the simulations show that the influence of the sampling frequency on the response time is negligible.

4.3.2 Impact of feature frequency

Figure 4 shows the performance results at varying feature frequencies for both correlated and uncorrelated movement ($f_s = 40$ Hz and $H = 5$ s). Beyond 4 Hz, the variances do not improve much anymore and the mean correlation value of correlated motion starts decreasing. We assume that this is caused by the fact that more high-frequency motion components are involved in the correlation when the feature frequency is higher, increasing the chances of mismatches between the signals.

4.3.3 Impact of correlation history

The top plot of Fig. 5 shows the accuracy variation with the correlation history length ($f_s = 40$ Hz and $f_f = 4$ Hz). For correlated movement, the optimum mean and variance are reached for a history length of 2 s. The performance for uncorrelated movement is more dependent on the history length, however, as the variance keeps decreasing until a history length of 8 s. This asymmetrical behavior is to be expected since uncorrelated movement usually has short coincidental periods in which the movement correlates, briefly producing a high correlation value when this period is shorter than the history length. A higher history length thus considerably reduces the chances of false positives.

The impact of the correlation history length on the response time is shown in the bottom plot of Fig. 5. Up to about 3.7 s, the response time for correlated and uncorrelated movement is very similar, but after that, a difference between the response time slopes is noticeable. As expected, the plot shows that faster response times can be achieved with a shorter correlation history. However, this will negatively impact in the accuracy, as explained above.

4.3.4 Impact of correlation thresholds

The decision of whether the sensors are moving together or not is based on comparing the correlation value to predefined thresholds. We use the simulation results to analyze the impact of the thresholds on the overall performance.

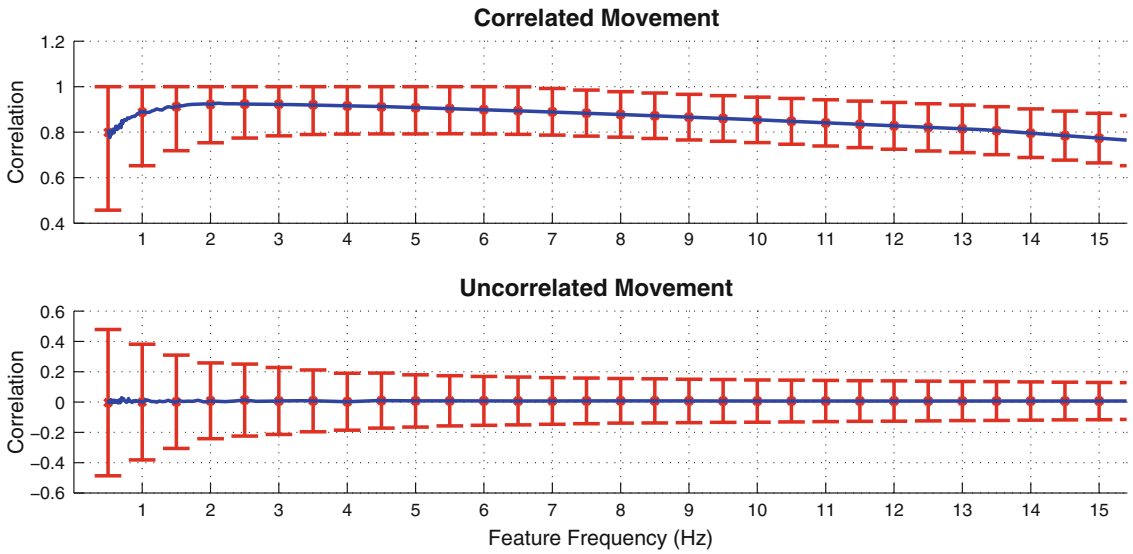

Fig. 4 Correlation output statistics at varying feature frequency

The results are illustrated in Table 2 ($f_s = 24$ Hz, $f_f = 4$ Hz and $H = 3$ s). The error percentages are obtained by counting the fraction of time the detector produces false correlation and false non-correlation results with the given thresholds. We notice the merit of using different thresholds for the transitions from and to correlation, yielding a hysteresis between the two assessments. The configuration with thresholds 0.65/0.45 provides the best trade-off between response time and accuracy.

4.3.5 Conclusion of simulation results

Given the presented simulation results and the identified trade-offs, we choose to set the sampling rate to 24 Hz, the feature frequency to 4 Hz, the correlation history length to 3 s and the correlation thresholds to 0.65/0.45. With these settings, the accuracy is close to optimal, and the upper limit of 2 s we set for the typical response time is still feasible. These are the settings we use for the hardware experiments outlined in Sect. 7.

5 Comparison of algorithms

In this section, we compare the performance of several alternative motion correlation algorithms, including two described in the work by Fujinami et al. [10, 11]. We first compare the performance in terms of accuracy and response time, and we conclude this section with a comparison of the required computational effort for these algorithms.

From our work, we include in this comparison the correlation ρ_{mam} of the acceleration-based f_{mam} feature (Equation 2), the correlation ρ_{cra} of the compass-based f_{cra} feature (Equation 1) and the combination ρ_{combined} of these correlations as described in Sect. 3.4. The performance of ρ_{mam} and ρ_{cra} is assessed separately to determine to what extent the combination ρ_{combined} is better than the correlation of one of these features alone.

Several accelerometer-based algorithms are discussed in the initial paper by Fujinami et al. [10], but we limit ourselves to the two most successful in this comparison: *raw-compo* and *raw-max*. Also, the work by Fujinami et al. describes using multiple sensors spread over the body and choosing the best one to correlate with the artifact used. For this comparison, we assume that this step is already taken and that the potentially associated pairs are known.

The work by Fujinami et al. gives no indication that any kind of gravity compensation is performed for the *raw-compo* or *raw-max* algorithms, so we will use the raw accelerometer data for these. Also, these algorithms use no feature extraction and are thus calculated for each individual sample: the window shifts one sample each time, meaning that the overlap is at its maximum. The correlation result is obtained by calculating the correlation directly over all the recent samples within the defined window size W. Contrary to our work, the absolute value of the correlation result is used. According to the initial publication [10], this is done to account for situations in which the nodes have opposing orientations and thus yield anti-correlated motion signals. This is thus part of their approach to address the orientation dependencies we discussed in Sect. 3.1.

The *raw-compo* algorithm uses the magnitude $|\mathbf{a}|$ of the acceleration signals $\mathbf{a_A}$ and $\mathbf{a_B}$ to correlate the movements between the nodes A and B in the sample window W. Much

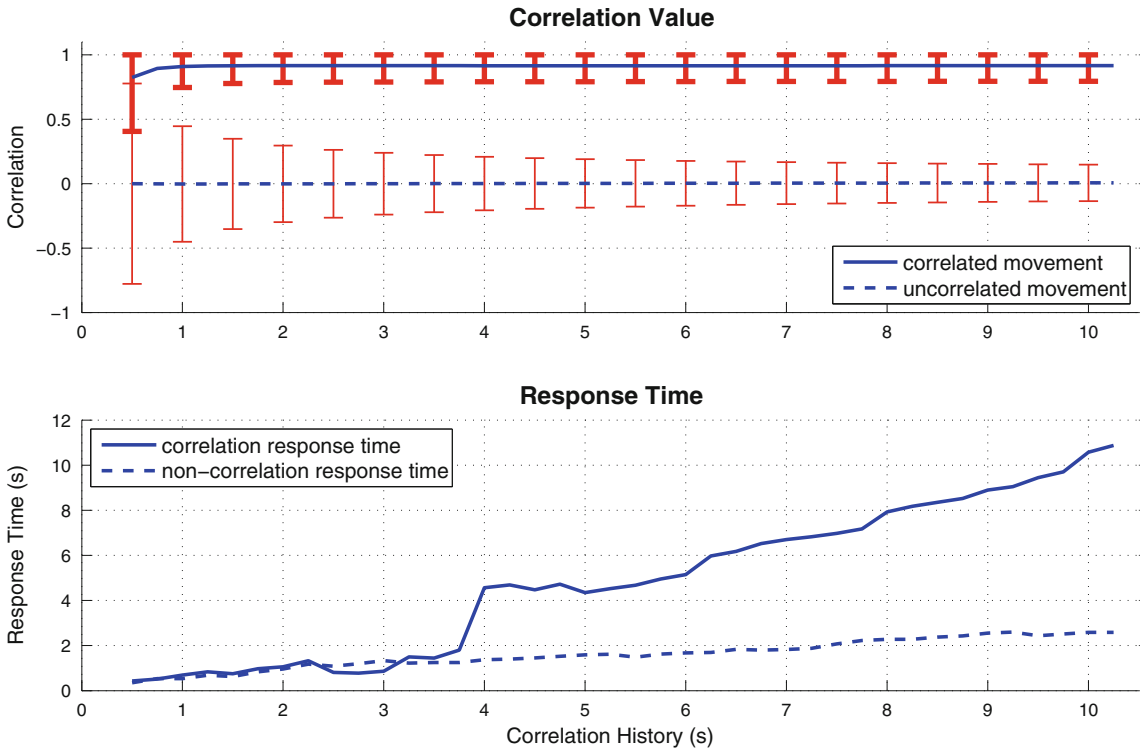

Fig. 5 Correlation output and response time statistics at varying correlation history lengths

Table 2 Correlation threshold statistics

Thresholds		Response time (s)		Errors (%)	
Corr.	Uncorr.	Corr.	Uncorr.	Corr.	Uncorr.
0.30	0.30	0.69	0.57	1.56	11.50
0.50	0.50	0.75	0.61	1.76	2.84
0.70	0.70	1.02	0.59	3.83	0.64
0.90	0.10	2.40	0.57	1.54	0.05
0.75	0.25	1.20	0.57	1.55	0.43
0.60	0.40	0.94	0.57	1.59	1.27
0.65	0.45	0.94	0.61	1.64	0.89

like our ρ_{mam} algorithm, the magnitude is used to obtain a scalar signal that retains only the intensity of the motion, thereby reducing the effects of differences in sensor orientation. We have implemented the *raw-compo* algorithm as follows using the sample correlation coefficient from Eq. 4:

$$\begin{aligned} |\mathbf{a}| &= \sqrt{a_x^2 + a_y^2 + a_z^2} \\ \rho_{\text{rcompo}} &= |\text{corr}_s(|\mathbf{a_A}|, |\mathbf{a_B}|, W)| \end{aligned} \tag{6}$$

The *raw-max* algorithm performs a cross-correlation over the sample window W for all nine axis pairs between the two nodes and selects the highest correlation value as the result. If the movements of the sensors are correlated, it is very likely that at least one axis pair in the cross-correlation will have significant (anti-)correlation. This is an alternative approach to compensate for differences in accelerometer orientation. We have implemented the *raw-max* algorithm as follows:

$$\begin{aligned} \rho_{i,j} &= |corr_s(a_{A,i}, a_{B,j}, W)| \\ \rho_{\text{rmax}} &= \max\{\rho_{x,x}\rho_{x,y}\rho_{x,z}\rho_{y,x}\rho_{y,y}\rho_{y,z}\rho_{z,x}\rho_{z,y}\rho_{z,z}\} \end{aligned} \tag{7}$$

5.1 Accuracy

First, we compare the accuracy of the algorithms. For this comparison, we need to define how the accuracy of the discussed algorithms is to be assessed. The work by Fujinami et al. assesses the accuracy in terms of correct association with one of two [10] or more [11] persons. In real situations, however, a second person to compare the correlation to is not always (if not rarely) present. In the more recent publication [11], the single-user situation is addressed by first matching the object's use state and the user's activity. This allows skipping the correlation calculation entirely, if only one user is engaged in an activity that matches the object's current state. The user is then assumed to be the one interacting with the object. Unfortunately, this fails to address the situation in which the object may not be actively moved by a user at all, e.g., it can be moved by some other person not involved in the system. Also, the object's status may not explicitly be part of the user's activity state, as Fujinami et al. [11] explain

with an example involving a coffee cup. Furthermore, this solution depends on knowledge of the user's activity and the object's use state, which is not always available nor easy to obtain. Therefore, we deem such relative accuracy metric inadequate for this comparison.

A good correlation method yields a correlation result close to the maximum (1.0) for perfectly correlated motion and a result close to zero for uncorrelated motion. When this is achieved, the use of a threshold-based method for the association decision is feasible, avoiding the need for comparison with other candidates. For optimal performance in terms of accuracy and reliability, we need a correlation method with a *high separation* between the values produced for correlated and uncorrelated motion. The best separation of 1.0 is achieved when correlated motion consistently yields a value of 1.0 and uncorrelated motion yields a value of 0.0 or lower. The separation is one of the accuracy metrics we use in this performance comparison, in addition to the mean and variance metrics explained earlier in Sect. 4.1.

We perform the comparison using the same data set as used in the performance evaluation simulations discussed in Sect. 4.2. For the comparison, we use a sample frequency f_s of 24 Hz, and for all algorithms, the correlation is performed over a total window (history) of 3 s. These values not only match the optimum we determined for our algorithm in Sect. 4.3, but also closely match the settings used in the work by Fujinami et al., originally being 17 Hz and 2.9 s (50 samples), respectively. The feature frequency used for our algorithm is 4 Hz.

Table 3 shows the statistics gathered from the accuracy simulations. The mean and the standard deviation of the produced correlation values are shown for each feature for correlated and non-correlated motion. Figure 6 provides a visual overview, with the bars displaying the mean values and the error bars displaying the standard deviations around these values. Since the correlation algorithms by Fujinami et al. use the absolute value of the correlation coefficient which therefore cannot be negative, the negative values from our correlation methods are all mapped to 0.0 for proper comparison.

Table 3 Comparison of movement correlation statistics

	Mean			SD	
	Corr.	Uncorr.	Sep.	Corr.	Uncorr.
ρ_{rcompo}	0.50	0.15	0.35	0.13	0.12
ρ_{rmax}	0.97	0.28	0.69	0.06	0.16
ρ_{mam}	0.88	0.13	0.75	0.11	0.18
ρ_{cra}	0.94	0.12	0.82	0.09	0.18
ρ_{combined}	0.91	0.10	0.81	0.09	0.14

The mean correlation produced by raw-compo is only 0.5 for correlated motion. This is due to the rotary motion influence discussed in Sect. 3.1. This is apparent when this result is compared to the statistics of the f_{mam} feature, which is—apart from coarse gravity compensation—essentially very similar to raw-compo: It correlates much better with a mean of 0.88. The separation of the raw-compo algorithm is very minimal at 0.35, which is by far the lowest value. When the standard deviation is also considered, the detection of simultaneous movement becomes very unreliable. This is apparent from Fig. 6, where the tips of the error bars for raw-compo are very close together.

For correlated motion, the raw-max algorithm stands out with a mean value very close to 1.0 for correlated motion. When two triaxial accelerometers move together, it is very likely that at least one pair of their axes will correlate significantly, explaining this very good result. However, with uncorrelated motion, raw-max has a mean value that is significantly higher than 0.0. Taking the standard deviation of 0.16 into account, values produced by raw-max can easily be as high as 0.44 for uncorrelated motion. The separation is also much lower than 1.0. Choosing the maximum correlation value from all axis pairs performs well for detecting correlated motion, but it performs poorly for uncorrelated motion, since the largest coincidental correlations will determine the result.

Table 3 shows very good results for the compass-based f_{cra} feature. The mean value for both correlated and uncorrelated motion is better than the accelerometer-based

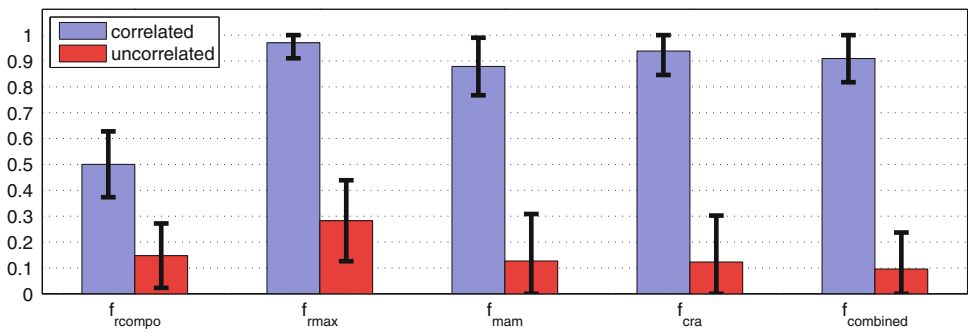

Fig. 6 Visual comparison of movement correlation statistics

f_{mam} feature. The mean value for correlated motion is even better than their combination, and the separation is higher than any other algorithm, suggesting that the compass sensor alone can perform very well. However, combining these features does have a clear advantage: The mean value for uncorrelated motion is lower than both individual features, and the standard deviation improves as well.

5.2 Response time

We also evaluate how the algorithms perform for the response time experiments explained in Sect. 4.2. For these simulations, it is necessary to define thresholds for the transitions between correlated and uncorrelated motion (the association decision). Considering the results of Table 3, it is not fair to choose these thresholds equal for each algorithm: The mean values for correlated and uncorrelated motion are not equal, and their standard deviations differ as well.

To keep matters simple, we use an identical threshold for both transitions between the correlated and uncorrelated state. We still need to choose a good threshold for each algorithm, which should lie somewhere between the mean values for uncorrelated and correlated motion shown in Table 3. It is also important to consider the standard deviations, such that the threshold is proportionally farther away from the decision with the largest deviation. To determine the threshold, we assumed the correlation outputs for correlated and uncorrelated motion to be of Gaussian nature with the parameters shown in Table 3, and we determined the threshold from the intersection point of the two Gaussian curves using the formula presented in [18].

Table 4 shows the results. The used threshold is shown for each algorithm. In addition to the mean response time measured during these simulations, the fraction of time the algorithms produced erroneous results is also compared. No motion detection is performed, so the response time and error results depend fully on the correlation algorithms themselves.

The performance of raw-compo is the worst, with the longest response times and very high error percentages. This is consistent with what the accuracy experiments show. The raw-max algorithm performs much better. Particularly, the error percentage for correlated motion is the lowest of all algorithms. However, the response time and error percentage for uncorrelated motion are still relatively high, which is also consistent with the accuracy experiments.

The accelerometer correlation ρ_{mam} is in terms of response time better than the compass correlation ρ_{cra} for uncorrelated motion, but worse for correlated motion. The combination ρ_{combined} finds a middle ground between the two with very good performance. Something similar is true for the error percentages, but in this case, the combination is consistently better than both individual features. Our combined algorithm is also better than the raw-max algorithm by Fujinami et al. except for the error percentage for correlated motion. With uncorrelated motion, the raw-max algorithm performs much worse.

Table 4 Comparison of algorithm response time performance

	Threshold	Response time (s)		Errors (%)	
		Corr.	Uncorr.	Corr.	Uncorr.
ρ_{rcompo}	0.32	3.03	2.32	31.39	12.18
ρ_{rmax}	0.77	2.04	1.18	0.43	4.08
ρ_{mam}	0.58	2.18	0.82	2.58	3.67
ρ_{cra}	0.65	1.41	1.03	2.91	2.60
ρ_{combined}	0.59	1.68	0.88	1.29	1.47

5.3 Computational effort

Since we aim for an online system, the correlation algorithm needs to run on the wireless sensor node hardware. The computational capabilities of sensor node hardware are usually quite limited. If the required effort is high, the node may not be able to finish the computations in time. In fact, the association module should only occupy a small fraction of the system resources, as this would in practice only be a subcomponent of an activity recognition system. Also, since computational effort adds to the energy consumption of the device and wireless sensor nodes operate on batteries, computational effort directly impacts battery life. For these reasons, the required computational effort should be as little as possible.

The algorithms by Fujinami et al. are executed for each accelerometer sample, meaning that the correlation coefficient is calculated at the sample frequency using raw samples. In contrast, our methods use signal features which are produced less frequently. This not only means that less correlation values need to be calculated per unit of time, but also that each correlation calculation is computationally less involved because less values need to be correlated in the same period of time: The number of values inside the correlation window/history is smaller. Our algorithms need to calculate correlation values 4 times per second over a history of 12 feature values, while the algorithms by Fujinami et al. need to calculate correlation values 24 times per second over a window of 72 samples. Even though our combined algorithm needs to calculate separate correlation values for the compass and accelerometer features, it is still less computationally intensive than either of the algorithms by Fujinami et al. The raw-max algorithm by Fujinami et al. is even worse in this respect, because it needs to calculate nine cross-correlation values at each instance.

Table 5 Comparison of computational effort required per second

	Computations per second		
	Additions	Multiplications	Square root
ρ_{rcompo}	15,504	952	48
ρ_{rmax}	139,296	7,920	216
ρ_{mam}	612	176	4
ρ_{cra}	448	216	12
ρ_{combined}	1,076	404	16

For a more concrete comparison, Table 5 shows the number of additions, multiplications and square root functions that need to be evaluated per second for each algorithm. These figures not only include the correlation calculations, but also feature computation and overhead such as compass calibration.

5.4 Conclusion

The performance of the raw-compo algorithm is very poor in this comparison. We attribute this mostly to the gravity effects for which this algorithm has no compensation. The good results in the original work by Fujinami et al. [10] are explained by the relative accuracy assessment method used in those experiments. The raw-max algorithm performs much better. Choosing the highest absolute correlation value among all nine axis pairs performs very well for correlated motion. However, when subjected to uncorrelated motion, using the largest (coincidental) correlation from all nine pairs yields relatively poor results.

It is clear that the correlation of the compass signal is a very good means to assess simultaneous motion. It performs better than the accelerometer-based algorithms in most cases, probably mainly because it lacks the gravity effects that hamper the performance of the accelerometer correlation. Although the correlation of the compass feature performs very well on its own, the combination of both compass and accelerometer is a significant improvement still. The additional sensor that measures a different aspect of the motion—the rotation instead of only linear motion—improves the accuracy and the reliability of the result, mainly because coincidental correlation is less likely to occur.

In terms of computational effort, the raw-compo and raw-max algorithms are significantly more expensive than the approaches proposed in this paper. We attribute this mainly to the fact that the raw-compo and raw-max algorithms compute the correlation for each sample, which means that this computation is performed more frequently over a longer window of values. In addition, the raw-max algorithm is more expensive because all nine axis pairs need to be correlated.

6 Implementation

This section outlines the specifics of our implementation.

6.1 Hardware

We use the ProMove [14] wireless inertial sensor nodes for this work. The ProMove board (Fig. 7) features a 3-D accelerometer and a 3-D digital compass. The main CPU of the sensor node is a low-power MSP430 microcontroller [29] running at 8 MHz. The nodes can communicate wirelessly using a CC2430 SoC [28], which combines an IEEE 802.15.4-compatible radio with an 8051 CPU. The CC2430 CPU autonomously handles the wireless networking. The ProMove architecture thus allows implementing an application in a two-tiered manner: performing data processing on the MSP430 and wireless networking on the CC2430.

6.2 Software

Figure 8 shows an overview of the software components involved in our implementation for a pair of nodes. Both nodes process the raw signals from their accelerometer and compass sensors into window intervals and calculate features from these intervals. This reduces the data rate and dimensionality, yielding a feature vector with only two values, as explained in Sect. 3.2. Subsequently, the movement correlation between the two nodes is determined. The feature vector of one node is communicated wirelessly to the other node, which performs the correlation calculation. The accelerometer and compass features are correlated separately, yielding two distinct correlation values. The final stage in the process, the *decision logic*, combines the two correlation values into a discrete interaction assessment.

The nodes exchange the necessary correlation messages wirelessly using the IEEE 802.15.4 protocol. One node acts as the coordinator and broadcasts its feature vector to slave nodes, which check whether they are moving together with the coordinator.

The sampling and feature extraction tasks running on the slave nodes need to be synchronized to the coordinator for proper correlation performance. For our experiments, a very simplistic synchronization procedure is implemented. It is executed only in the system startup phase, and the system is restarted for each experiment to ensure proper synchronization. All nodes maintain a local sample counter that increases for every sample taken. When a new slave node is added to the network, the coordinator sends its local sample counter value to the new node. The new node uses the message to adjust its local sample counter and echoes the message back to the coordinator at the time of

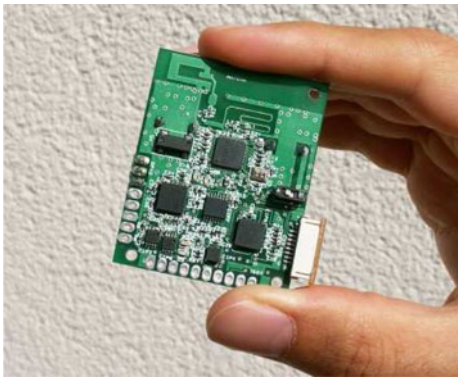

Fig. 7 ProMove inertial sensor board

the next sample. The coordinator uses the reply to calculate the counter difference in terms of samples, which is then communicated back to the slave node along with the new current counter value. Using this second message, the slave node updates its local sample timer using the counter value from the master and half the communicated difference. These steps are repeated until the coordinator receives the next reply at the moment its local counter value is equal to the sum of the counter value and the difference value contained in that reply. This achieves coarse synchronization within one sample. The feature extraction is synchronized once the sample counters are synchronized.

6.3 Benchmark

To investigate the feasibility of our implementation, we run a benchmark to measure the processing load on the MSP430 processor. Table 6 lists the results for each task. The correlation computation, sensor sampling and feature extraction operations have the longest execution times, as expected. The complete implementation uses only approximately 7% of the processor's time, leaving therefore ample resources available for the high-level application.

The energy usage of the implementation is currently not optimized, e.g., CPU and radio sleep modes are not employed. Most of the energy is spent in the communication, which is performed at a rate of 4 Hz, i.e., the feature frequency. Varying the configuration of the correlation algorithm is barely noticeable in the power consumption. The power consumption measured for the full prototype is currently about 150 mW per node.

7 Tests and results

To evaluate our implementation, we perform a series of experiments with handling objects equipped with our sensors. In each experiment, the user wears one sensor node on a bracelet on his arm and two other sensor nodes are placed onto or inside objects, as shown in Fig. 9. The arm node acts as the protocol coordinator and the usage detection is performed in the object nodes. The exchanged feature vectors and the resulting assessments are logged by a PC with a gateway node for later evaluation. In these experiments, the nodes are less synchronized compared to the offline evaluation (within one sample instead of microsecond range), and there is no compensation for the potential packet loss.

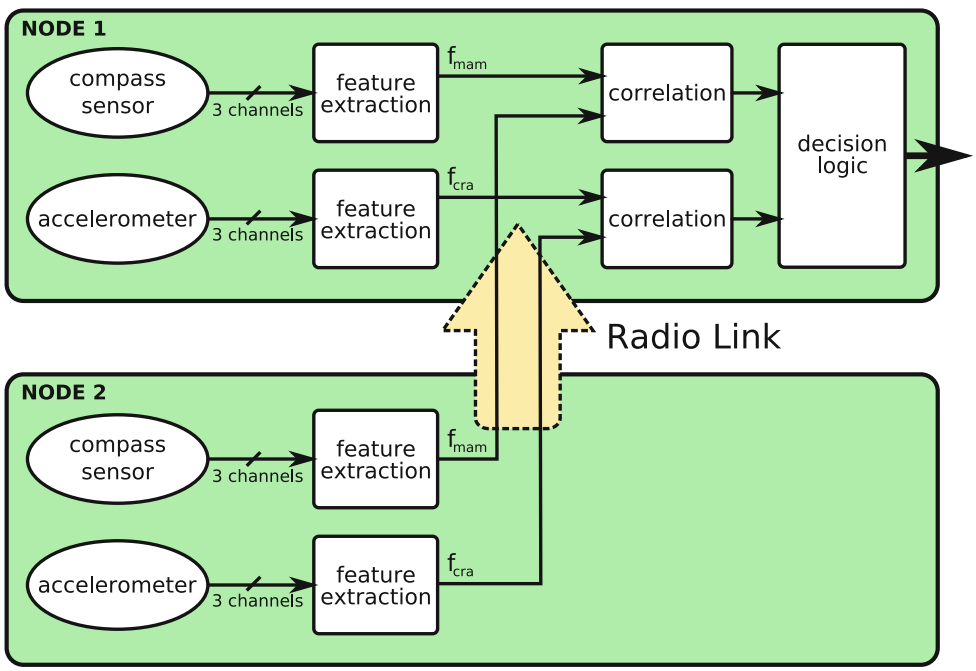

Fig. 8 Software overview

Table 6 Software benchmark results

Subsystem	CPU time	
	Cycles (8 MHz)	Time (ms)
Sampling (24 Hz)	166,248	20.78
Windowed feature extraction (4 Hz)	147,644	18.46
Correlation (4 Hz, 3 s history)	209,360	26.17
Communication (4 Hz)	11,556	1.44
Overhead (I/O wait, timeouts, etc.)	2,459	0.30
Total system load	537,267	67.16

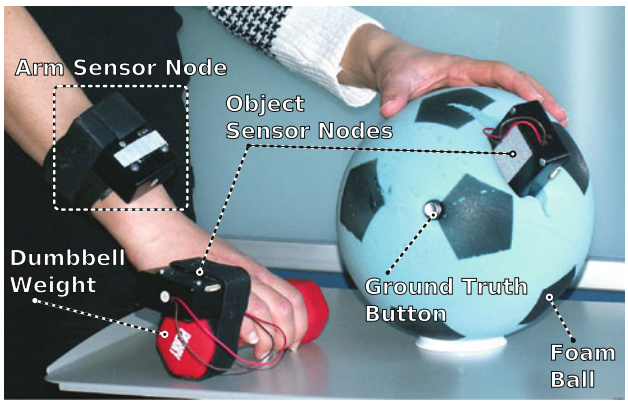

Fig. 9 Some of the hardware involved in the implementation experiments

7.1 Generic movement

First, we assess the response time and accuracy of the algorithm when correlating generic movement of an object held by the user. We let our test subjects perform random motion with the objects, i.e., any motion they see fit. We use two foam balls (one of which is depicted in Fig. 9) that can be handled in any orientation. The user can handle one of the balls with the arm on which he wears the sensor. A second person moves the balls that are not currently held by the user, thus trying to generate false correlations. The user is not necessarily always moving one of the two balls, in which case the second person may move them both. Balls can be placed still on the table during the experiment for the other person to pick them up but also handed over directly. We experiment with ten different users, which perform five individual tests. Each of these 50 individual tests lasts two minutes.

Similar to the off-line evaluation, push buttons on the balls are used for automated annotation of the ground truth. While grabbing and holding one of the foam balls, the user with the bracelet keeps the ground truth button on that ball pressed continuously. The second person does not touch the buttons at all.

Figure 10 shows an example experiment, comparing the true object association as indicated by the button (solid black lines) and the output of our detection algorithm (dashed red lines). The produced correlation values are shown as well. The two plots show the object use association results for the two foam balls. Approximately at 4 s, both balls are picked up from the table and start moving. Ball 2 is held and moved by the user with the bracelet, while Ball 1 is moved by the second person. At 27.5 s, the user hands over Ball 2 to the second person, who at that point moves both balls at the same time. As shown in the graph, no ball is associated with the user at that time. A little later, at 47 s, Ball 2 is handed back to the user. At 65 s, the balls are swapped between the user and the second person, which is the first time that Ball 1 is held by the user. At 88 s, the balls are swapped back. Just before the experiment finishes, both balls are placed back on the table.

Table 7 shows the overall performance of our implementation for all 50 tests. As expected, in most cases, the performance is worse than in the simulation. However, the response times are typically within the 2 s limit, and the accuracy of the algorithm is adequate, with false correlation at about 3% of the time and false non-correlation at about 2% of the time. There is one user that exceeds the 2 s response time by about half a second. The reasons for this outlier are not known.

7.2 Activities

The aim of this research is to devise a means to establish a usage relation between an object and a user by comparing the movement of both. In the previous section, we tested the performance of our algorithm in the more generic scenario where the user is holding the object and moving it around. In the experiments outlined in this section, the objects are used in actual defined activities. We chose four activities that were relatively easy to verify: exercising with a dumbbell, wiping a whiteboard, using a hammer and painting with a brush. Each test involves two persons performing the same activity, of which one wears the bracelet. The objects are the tools used in the activities, each equipped with sensors, as shown in Fig. 11. The dumbbell and whiteboard activities are performed in the office, whereas the hammer and brush activities are performed in a workshop environment. Each activity is tested by a total of three persons and performed six times by alternating pairs. The duration of each test is one minute. The objects are always picked up after ten seconds and laid down ten seconds before the end of each test, so no ground truth buttons are used in these experiments.

Table 8 shows the performance results. The whiteboard and dumbbell activities show excellent performance. The response time is about one second for both the transition to

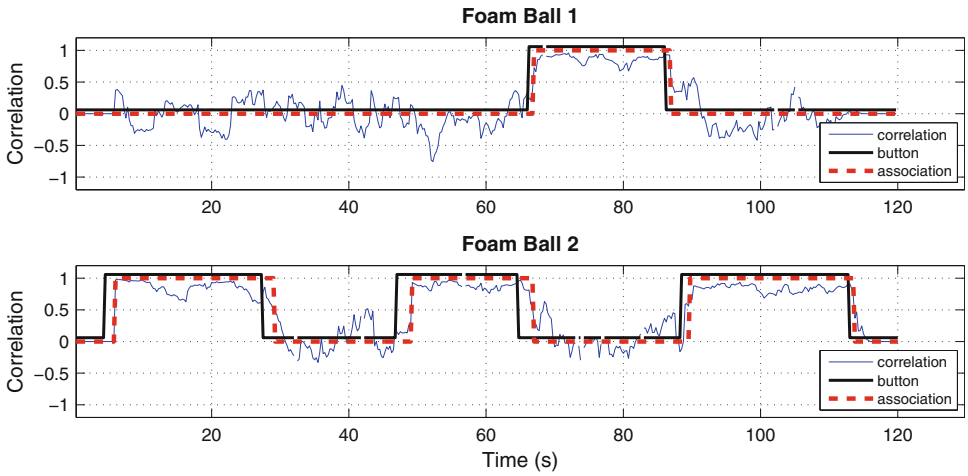

Fig. 10 Example of an experiment with the implemented system

Table 7 Implementation performance statistics

	Response time (s)		Errors (%)	
	Corr.	Uncorr.	Corr.	Uncorr.
User 1	1.61	1.41	4.44	2.05
User 2	1.66	1.48	0.96	2.17
User 3	1.41	2.00	0.37	2.37
User 4	1.73	0.76	5.77	2.39
User 5	1.23	1.59	3.44	0.53
User 6	1.82	0.89	6.66	1.04
User 7	1.21	1.21	5.45	3.87
User 8	1.05	1.86	2.81	1.80
User 9	1.20	2.28	0.68	2.48
User 10	1.23	1.95	0.40	1.14
Mean performance	1.41	1.55	3.10	1.98
SD	0.27	0.49	2.41	0.94

correlating and the transition to non-correlating movement. The errors are limited to little over 1%. For the dumbbell experiments, the system never failed to identify correlation, while the system failed to do so for the whiteboard experiments 1% of the time. The system falsely reports correlation 1.1% of the time for the dumbbell experiments and 0.5% for the whiteboard experiments. It should be noted that the response times are very low because the objects are not moving when not used, causing the motion variance detection scheme to reduce response time, as explained in Sect. 3.5.

As shown in the table, the results for the workshop experiments with the brush and the hammer are less ideal. The system still assesses the situation that the movement is not correlating and that tool is thus not being used with high accuracy. However, the system is less capable of correctly assessing the use of the tools. This is probably related to the fact that these two activities involve more wrist mobility than the whiteboard and dumbbell activities. Particularly, the rotary motion will correlate less for such activities, which suggests that the way accelerometer and compass correlation values are currently combined is not always ideal; for some activities, the correlation of the accelerometer can be more reliable than the compass correlation. Also, the workshop environment may have contributed to the less ideal results, since much ferrous metal is in close proximity there, which may have influenced the compass sensor.

8 Interactive ball game

To illustrate the potential of our motion-based interaction detection method in a more practical scenario, we implement a prototype for a simple interactive ball game with multiple players. The game is a variant of the "Hot Potato" game. In the original game, the players gather in a circle and toss around a ball while music plays. The player who holds the ball when the music stops is out, and the game continues with a new round until only the winner remains.

In our implementation of the game, each player has a sensor node attached to one arm. A sensor node is also embedded in the ball. When receiving the ball, a player first has to move it for a short while using his arm with the sensor node, so that correlation is achieved; then, he can pass the ball to another player. The graphical interface of the game is depicted in Fig. 12. Each player has a smiley avatar, which disappears when the player is out. The ball is shown as a cloud of sparkles surrounding the player who handles it. This indicator jumps to another player when the ball is tossed.

Figure 13 shows three players involved in the game. All three players are still in the game and the ball is just being passed. The sensor node in the ball is the coordinator in the wireless communication protocol. The sensor nodes on the arms of the players continuously determine their movement

Fig. 11 Performed activities

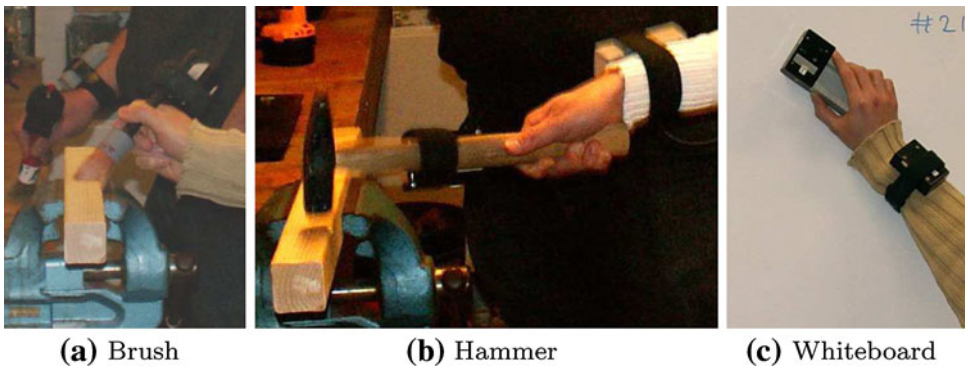

Table 8 Activity performance statistics

	Response time (s)		Errors (%)	
	Corr.	Uncorr.	Corr.	Uncorr.
Dumbbell	1.04	0.94	0.00	1.11
Whiteboard	1.00	0.94	1.09	0.46
Hammer	1.63	0.77	7.79	0.00
Brush	1.69	0.56	9.13	0.90

correlation with respect to the coordinator. Each node on a player's arm broadcasts its correlation value four times per second. These broadcasts are picked up by a gateway node which is connected to the computer that runs the game graphical interface. The game is projected on the large screen behind, where we notice the ball being passed between players. In addition to the screen interface, the players can also see when they are associated with the ball by means of LEDs on the arm sensors.

The game proves to be a very entertaining experience for the users. The requirement of first moving or shaking the ball before tossing it is perceived as an interesting addition to the original "Hot Potato" game, stimulating the user—object interaction and the physical activity level. The correlation accuracy is satisfactory from a gaming perspective, with correct recognition of ball possession in almost all cases. The transition between players is however a point of further improvement.

The statistics we collected from ten games indicate an average response time of 2.6 s with a standard deviation of 1.3 s. A histogram of the response time of each instance the ball was tossed during the experiments is shown in Fig. 14. The response time is larger than that achieved in our earlier experiments. The main reason is the additional delay introduced by the communication that needs to take place between the player nodes and the gateway, on the one hand, and between the gateway and the computer, on the other hand. Occasional packet loss is also a factor here. Thus, even though the response time on the actual sensor nodes will typically remain within the 2 s target, the reaction of the graphical interface is somewhat slower.

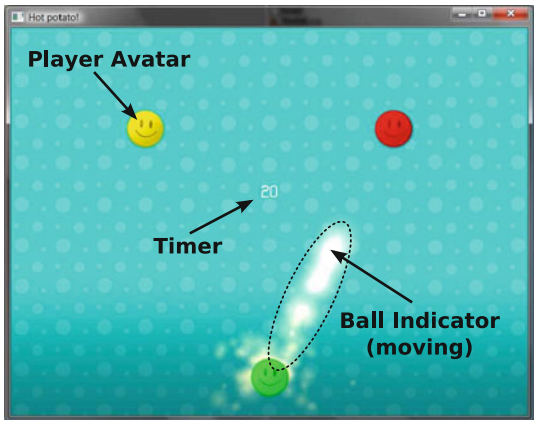

Fig. 12 A screenshot of the Hot Potato display

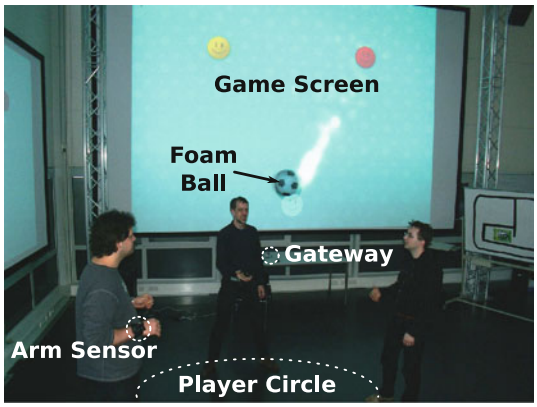

Fig. 13 People playing the Hot Potato game with screen in the background

9 Discussion and conclusions

We presented a method for automatic recognition of object use, based on correlating motion features in a collaborative manner among sensor nodes attached to the user's arm and to the handled objects. Being based on motion sensing, our solution provides information about the actual usage of objects instead of only the proximity of the user to the object. Also, our solution detects *which* objects the user is

interacting with, while also offering the possibility to infer *how* the objects are used. More specifically, since we perform both feature extraction and feature correlation, the outputs of these building blocks could be used directly to implement distributed activity recognition. Furthermore, the method we propose is generic and can be applied to build associations of the type "moving together" for any entities equipped with sensors. It is therefore not restricted to a particular one-to-many or many-to-one interaction scenario. And finally, we prove that our solution can run on resource-constrained hardware, taking only a fraction of the CPU time and operating using the 802.15.4 MAC protocol, which is suitable for wireless sensor networks [2].

In the remainder of this concluding section, we refer back to the research questions formulated in Sect. 1 and outline the answers given by our study.

What are the signal features that express well and compact the motion information, while remaining computationally simple enough to be implemented on resource-constrained hardware? In Sects. 3.1 and 3.2, we explain how the accelerometer and compass sensors can be used to extract features that characterize well both linear and rotary motion components. The selected features are the compass rotation angle and the mean acceleration magnitude. They match the set requirements: retain the overall motion characteristics, are computationally efficient, have low feature size and rate and do not depend on the sensor orientation. In Sect. 5, we evaluated the merit of combining the compass and accelerometer features. The compass feature alone already performs better than the accelerometer feature, but their combination is still a clear improvement: Particularly, in terms of accuracy and reliability, the combination is better, which is mainly visible for the detection of uncorrelated movement.

How can the correlation be done efficiently among the sensor nodes? Section 3.4 describes the correlation algorithm and the heuristics used for combining the results from the accelerometer and compass sensors. The algorithm is based on the correlation coefficient, which gives a good estimation of the motion similarity of two or more entities, while being computationally efficient to implement on sensor nodes. Furthermore, Sect. 3.3 details the dependencies between the correlation algorithm and the communication and synchronization of features.

What are the relevant parameters and trade-offs? How to choose the optimal values? The algorithm parameters are identified in Sect. 3.6: sensor sampling rate, feature frequency, correlation history length and decision threshold. In order to establish the performance trade-offs and choose the optimal values, we perform a series of real experiments from which we wirelessly collect the raw sensor data at high rates. Next, we analyze the data off-line, varying each of the identified parameters individually while the others are held constant. The performance is measured in terms of detection accuracy and response time. With the chosen optimal values, the system achieves a maximum accuracy for a target response time of 2 s.

How does our algorithm compare to existing work? We compared our algorithms with earlier work by Fujinami et al. in Sect. 5. Our algorithms perform better in most respects. An important exception is that the raw-max algorithm by Fujinami et al. has higher accuracy and better response time for correlated movement. However, for detecting uncorrelated movement, it performs much worse. In addition to that, the computational effort of the algorithms by Fujinami et al. is much higher, mostly due to the fact that correlation values are calculated more frequently over a larger window of values. Also, the raw-max algorithm is even more expensive in this respect, since it calculates the correlation for nine axis pairs each time.

The difference between the performance results from Fujinami et al. [10] and ours is mostly explained by the difference in the chosen accuracy metric, as explained in Sect. 5.1. However, our comparisons are based on simulations using a relatively limited data set, which may have an influence on the accuracy and response time results. It would be better to have a much larger data set with raw sensor data from several people performing a large number of different activities.

How do we implement the overall system on sensor nodes? Section 6 presents the implementation on sensor nodes, covering the hardware details, software architecture

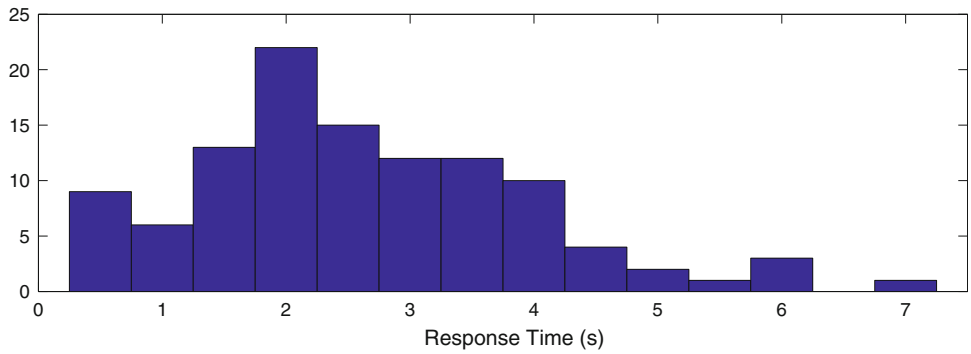

Fig. 14 Response time histogram from tests with Hot Potato game

and wireless communication. The feasibility of implementation is demonstrated by the detailed benchmarking given in Table 6.

What is the performance in practice, and how does it compare to simulations? We evaluate the performance through practical tests with the complete system working on sensor nodes. We distinguish three types of tests: generic handling of objects, actual activities involving object use and an interactive ball game. The detailed descriptions and performance results are given in Sects. 7 and 8. In terms of detection accuracy, we observe in general a good match between the implementation and simulation, with false correlation $\leq$3% and false non-correlation $\leq$2%. Some activities, like hammering or brushing, generate higher errors (7−9%) because they involve more wrist mobility that causes decorrelation. In terms of response time, the performance remains within the 2 s target for generic handling and activities and reaches 2.6 s on average in the game tests. These results indicate that the systems perform robustly in practice, with very slight performance degradation compared to the off-line evaluation.

What are the main problems, limitations and ideas for further improvement? The main problems and limitations of our solution and our ideas for their mitigation are summarized as follows:

- Some activities do not cause perfect and lasting motion correlation between the sensor in the object and the sensor on the user's arm. Especially activities involving significant wrist motion and activities where the object is not handled continuously with the instrumented hand will not yield continuous motion correlation between user and object. However, given the nature of the object, such characteristics can be taken into account. For example, when it is known that the typical use of a particular object can involve much wrist motion, the accelerometer correlation can be given more weight in the association result. Based on the object involved, the association algorithm could also allow short periods in which movement is less correlated. This would yield a more dynamic solution where the expected amount and type (i.e., rotary or linear) of movement correlation are directly linked to the activities that are possible with the objects involved. Also, the configuration of the association algorithm can be adjusted accordingly.
- Strong magnetic fields and large nearby ferrous metal surfaces affect the performance of the compass sensor. As a solution, a gyroscope could supplement or replace the compass as rotation sensor, thereby removing the sensitivity to magnetic disturbances.
- Correlation of linear motion does not perform well when there is also significant rotary motion. It can be beneficial for performance to be able to compare both rotary and linear motion at all times. To achieve this, gravity needs to be compensated properly. This requires information on the node's orientation, which can be inferred using the compass sensor or a gyroscope.
- It is inefficient to attempt motion correlation with distant objects. By incorporating radio signal strength (RSSI) information in the algorithm, nodes that are far apart can be omitted from the correlation process, which improves efficiency and scalability and additionally reduces the number of false positives.

References

1. Aylward R, Paradiso JA (2006) Sensemble: a wireless, compact, multi-user sensor system for interactive dance. In: Proceedings of the 2006 conference on New interfaces for musical expression. IRCAM Centre Pompidou, Paris, France, pp 134–139. URL http://portal.acm.org/citation.cfm?id=1142215.1142248
2. Baronti P, Pillai P, Chook VW, Chessa S, Gotta A, Hu YF (2007) Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards. Comput Commun 30(7):1655–1695. doi:16/j.comcom.2006.12.020, URL http://www.sciencedirect.com/science/article/pii/S0140366406004749
3. Bauer G, Stockinger K, Lukowicz P (2009) Recognizing the Use-Mode of kitchen appliances from their current consumption. In: Smart sensing and context, pp 163–176. URL http://dx.doi.org/10.1007/978-3-642-04471-7_13
4. Berlin E, Liu J, van Laerhoven K, Schiele B (2010) Coming to grips with the objects we grasp: detecting interactions with efficient wrist-worn sensors. In: Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction, TEI '10. ACM, New York, p 5764. doi:10.1145/1709886.1709898. ACM ID: 1709898
5. Bosch S, Marin-Perianu R, Havinga P, Horst A, Marin-Perianu M, Vasilescu A (2010) Automatic recognition of object use based on wireless motion sensors. In: Wearable computers (ISWC), 2010 international symposium, pp 1–8. doi:10.1109/ISWC.2010.5665858
6. Brunette W, Hartung C, Nordstrom B, Borriello G (2003) Proximity interactions between wireless sensors and their application. In: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications. ACM, San Diego, pp 30–37. doi:10.1145/941350.941356, URL http://portal.acm.org/citation.cfm?doid=941350.941356
7. Buettner M, Prasad R, Philipose M, Wetherall D (2009) Recognizing daily activities with RFID-based sensors. In: Proceedings of the 11th international conference on ubiquitous computing. ACM, Orlandopp. 51–60. doi:10.1145/1620545.1620553. URL http://portal.acm.org/citation.cfm?id=1620553
8. Feldman A, Tapia E, Sadi S, Maes P, Schmandt C (2005) ReachMedia: on-the-move interaction with everyday objects. In: Wearable computers, 2005. Proceedings. Ninth IEEE International Symposium, pp 52–59. doi:10.1109/ISWC.2005.44
9. Fujinami K, Nakajima T (2005) Sentient artefacts: acquiring users context through daily objects. In: Embedded and

ubiquitous computing, pp 335–344. URL http://dx.doi.org/10.1007/11596042_35
10. Fujinami K, Pirttikangas S (2007) A study on a correlation coefficient to associate an object with its user. In: Intelligent environments, 2007. IE 07. 3rd IET international conference, pp 288–295
11. Fujinami K, Pirttikangas S (2008) Kuka: an architecture for associating an augmented artefact with its user using wearable sensors. In: Sensor networks, ubiquitous and trustworthy computing, 2008. SUTC '08. IEEE international conference on, pp 154–161. doi:10.1109/SUTC.2008.42
12. Hinckley K (2003) Synchronous gestures for multiple persons and computers. In: Proceedings of the 16th annual ACM symposium on User interface software and technology, UIST '03. ACM, New York, p 149158. doi:10.1145/964696.964713. ACM ID: 964713
13. Holmquist LE, Mattern F, Schiele B, Alahuhta P, Beigl M, Gellersen H (2001) Smart-Its friends: a technique for users to easily establish connections between smart artefacts. URL http://eprints.comp.lancs.ac.uk/515/
14. Inertia Technology: ProMove wireless inertial sensor node. URL http://www.inertia-technology.com
15. Kawsar F, Fujinami K, Nakajima T (2006) Exploiting passive advantages of sentient artefacts. In: Ubiquitous computing systems, pp 270–285. URL http://dx.doi.org/10.1007/11890348_21
16. Kortuem G, Kawsar F, Fitton D, Sundramoorthy V (2010) Smart objects as building blocks for the internet of things. Internet Comput IEEE 14(1):44–51. doi:10.1109/MIC.2009.143
17. Lester J, Hannaford B, Borriello G (2004) Are you with me?—Using accelerometers to determine if two devices are carried by the same person. In: Pervasive computing, pp 33–50. URL http://www.springerlink.com/content/n22dwch7673kgkng
18. Marin-Perianu R, Lombriser C, Havinga P, Scholten H, Trster G (2008) Tandem: a context-aware method for spontaneous clustering of dynamic wireless sensor nodes. In: Floerkemeier C, Langheinrich M, Fleisch E, Mattern F, Sarma SE (eds) The internet of things, vol 4952, pp 341–359. Springer, Berlin. URL http://eprints.eemcs.utwente.nl/12250/
19. Marin-Perianu R, Marin-Perianu M, Havinga P, Scholten H (2007) Movement-based group awareness with wireless sensor networks. In: Pervasive computing, pp 298–315. URL http://dx.doi.org/10.1007/978-3-540-72037-9_18
20. Mayrhofer R, Gellersen H (2007) Shake well before use: authentication based on accelerometer data. In: Pervasive computing, pp 144–161. URL http://dx.doi.org/10.1007/978-3-540-72037-9_9
21. Patterson D, Fox D, Kautz H, Philipose M (2005) Fine-grained activity recognition by aggregating abstract object usage. In: Wearable computers, 2005. Proceedings. Ninth IEEE international symposium, pp 44–51. doi:10.1109/ISWC.2005.22
22. Philipose M, Fishkin K, Perkowitz M, Patterson D, Fox D, Kautz H, Hahnel D (2004) Inferring activities from interactions with objects. Pervas Comput IEEE 3(4):50–57. doi:10.1109/MPRV.2004.7
23. Schiele B, Antifakos S (2003) Grouping mechanisms for smart objects based on implicit interaction and context proximity. In: International conference on ubiquitous computing (UBICOMP 2003)
24. Smith J, Sample A, Powledge P, Roy S, Mamishev A (2006) A wirelessly-powered platform for sensing and computation. In: Dourish P, Friday A (eds) UbiComp 2006: ubiquitous computing. Lecture notes in computer science, vol 4206. Springer, Berlin, pp 495–506. doi:10.1007/11853565_29.10.1007/11853565_29
25. Strohbach M, Kortuem G, Gellersen H, Kray C (2004) Using cooperative artefacts as basis for activity recognition. In: Ambient intelligence, pp 49–60. URL http://www.springerlink.com/content/75bbmnxb3afphk33
26. Surie D, Laguionie O, Pederson T (2008) Wireless sensor networking of everyday objects in a smart home environment. In: Intelligent sensors, sensor networks and information processing, 2008. ISSNIP 2008. International conference, pp 189–194. doi:10.1109/ISSNIP.2008.4761985
27. Tapia EM, Intille SS, Larson K (2004) Activity recognition in the home using simple and ubiquitous sensors. In: Pervasive computing, pp 158–175. URL http://www.springerlink.com/content/5a4qm20y37089gk9
28. Texas Instruments: CC2430 System-on-Chip Solution for 2.4 GHz IEEE 802.15.4/ZigBee. URL http://focus.ti.com/docs/prod/folders/print/cc2430.html
29. Texas Instruments: MSP430 family of ultra-low-power microcontrollers. URL http://focus.ti.com/docs/prod/folders/print/msp430f1611.html
30. Wang S, Pentney W, Popescu A, Choudhury T, Philipose M (2007) Common sense based joint training of human activity recognizers. In: Proceedings of the 20th international joint conference on artificial intelligence, pp 2237–2242. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.61.5347
31. Wu J, Osuntogun A, Choudhury T, Philipose M, Rehg J (2007) A scalable approach to activity recognition based on object use. In: Computer vision, 2007. ICCV 2007. IEEE 11th international conference, pp 1–8. doi:10.1109/ICCV.2007.4408865