Automatic Recognition of Object Use Based on Wireless Motion Sensors

Stephan Bosch,
Raluca Marin-Perianu,
Paul Havinga, Arie Horst
University of Twente
Enschede, The Netherlands
s.bosch@utwente.nl

Mihai Marin-Perianu
Inertia Technology B.V.
Enschede, The Netherlands
mihai@inertia-technology.com

Andrei Vasilescu
PROSYS PC SRL
Bucharest, Romania
andrei.vasilescu@prosyspc.ro

Abstract

In this paper, we present a method for automatic, online detection of a user's interaction with objects. This represents an essential building block for improving the performance of distributed activity recognition systems. Our method is based on correlating features extracted from motion sensors worn by the user and attached to objects. We present a complete implementation of the idea, using miniaturized wireless sensor nodes equipped with motion sensors. We achieve a recognition accuracy of 97% for a target response time of 2 seconds. The implementation is lightweight, with low communication bandwidth and processing needs. We illustrate the potential of the concept by means of an interactive multi-user game.

1 Introduction

User interaction with smart objects represents an essential component of the ubiquitous computing vision. As debatable as the definition of a "smart object" may be, it is logical to assume that a smart object should, at the very least, detect when the user is interacting with it. A more advanced piece of functionality would be to additionally recognize how the user is interacting with it and further on, by combining such information from various objects, to infer the user's activities and provide specific support. For the basic problem of detecting what objects are handled by the user, the general approach is to use basic RFID or other strictly proximity-based solutions. This has however one important inherent limitation: the detection is solely based on the proximity of the user to the object and not on the actual usage. As a consequence, the system 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 [11].

In view of these problems, we investigate a different approach based only on motion measurements. We equip the objects of interest and the user's arm with wireless sensor nodes that measure their motion using inertial and magnetic sensors. In order to detect object use, we correlate relevant features of the collected motion data from the objects and the user's arm. The feature extraction, communication and correlation is done on-line and cooperatively by the sensor nodes, which guarantees a fast response time to the user.

Our solution accurately detects *what* objects the user is interacting with, while also offering the possibility to infer *how* the objects are used, without the need for additional hardware. Moreover, the method is generic and can be applied to build associations of the type "moving together" for any entities equipped with motion sensors. We describe and evaluate the *complete working solution*, fully implemented in sensor node hardware and tested with multiple users.

In this paper[1], we first survey previous work on detecting object use. Then, we describe our solution in detail and we tune and evaluate the performance of our solution using a series of simulations and experiments. Finally, we demonstrate the practical use in an interactive ball game.

2 Related Work

Performing or enhancing activity recognition using information on what objects the user is currently manipulating is already a well-established concept [11, 8]. For a single-user environment, the detection can be implemented by adding switches or other simple sensors to the points of interaction, such as doors, knobs and levers [10]. For a multi-user environment, it is also important to know *who* is using the object. Therefore, it is not sufficient to use a simple switch or contact sensor. A common solution is to employ basic RFID [11, 8] or other proximity-based solu-

[1]This work is sponsored by the SENSEI (http://www.senseiproject.eu) and IS-ACTIVE (http://www.is-active.eu) projects.

tions [1]. Unfortunately, proximity information alone is often not enough to achieve reliable object association, mainly because nearby unused objects can also be detected [11]. That is why we investigate the use of motion sensing for object association. Although this could be combined with the proximity based techniques and RFID technology, for instance using the sensor-instrumented WISP RFID tags [2], our research focuses on using motion sensor data alone.

Our solution to use simultaneous motion as a means for detection of object use is not a well-established concept. However, publications on detecting whether motion sensors are moving together exist for various other applications. Work by Lester et al. [5] 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. Also using frequency-domain analysis, Mayrhofer et al. [7] exploit shared movement patterns to authenticate communication between wireless devices. The user can pair the devices for secure communication simply by shaking them together. Unfortunately, frequency-domain analysis is relatively resource-intensive, making it less feasible for sensor nodes, and it will not perform well with non-repetitive movement, such as for a human arm manipulating an object.

For a transport and logistics application, Marin-Perianu et al. [6] describe a method to determine whether wireless sensor nodes attached to transportation items are moving together based on accelerometer data. Our application, however, has different requirements and challenges, because it involves human motion. The characteristics of human motion, compared to motion of transportation items, make the accelerometer-only solution less accurate. To improve accuracy, we base our solution on the fusion between the accelerometer and compass sensor data. We use correlation of motion features instead of raw sensor data, thus reducing significantly the overall communication requirements.

3 Solution Overview

Our solution is based on smart objects (also called sentient artifacts or cooperative artifacts [9, 4]), which envelop sensing, processing and communication capabilities. Each object is equipped with a wireless sensor node with 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 lateral movement, whereas the compass sensor measures rotary movement. 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.

3.1 Feature Extraction

The feature values are extracted from the raw signal at regular non-overlapping intervals called *windows*. A new 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 extracted features need to meet certain requirements: (1) 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, (2) the processing requirements need to be as low as possible, as dictated by the resource limits of sensor node hardware, (3) the extracted features must be small and produced at a low rate to keep bandwidth and processing requirements low, and (4) the features must be computed such that the absolute orientation of the sensor has little or no influence on the produced feature result.

To meet these requirements, we define a *feature vector* composed of two features that describe the intensity of lateral and rotational movements and are orientation-independent:

- *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 $\vec{m}(t) = \langle m_x(t), m_y(t), m_z(t) \rangle$ in the window interval $t = 1 \ldots N$ as follows:

$$f_{cra} = \frac{\vec{m}(1) \cdot \vec{m}(N)}{|\vec{m}(1)||\vec{m}(N)|} \quad (1)$$

  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 $t = 1 \ldots N$:

  1. The raw accelerometer signal $\vec{a_r}(t) = \langle a_{r,x}(t), a_{r,y}(t), a_{r,z}(t) \rangle$ is stripped from its offset by subtracting the mean value in the window interval.

This coarsely compensates for the gravity component, which is orientation-dependent and usually changes slower than the actual acceleration the sensor is subjected to. 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 $\vec{a_r}(t)$ in the window interval $t = 1 \dots N$ as follows:

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

3.2 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.

3.3 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:

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} \tag{3}$$

The correlation is calculated over the last H features produced at both nodes. The result $\rho_{X,Y}$ lies in the range $[-1; 1]$, for which a value of 1 means that the signals are fully correlated and values ≤ 0 (in our case) mean that the signals are not correlated.

Using Equation 3, separate correlation values are calculated for the two feature values. These results have to be combined into a single value that indicates how well the motion of the two nodes correlates. Preliminary experiments show that the reliability of the accelerometer correlation is sensitive to large 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_a and the compass correlation ρ_c. This done using the following heuristic formula:

$$\begin{aligned} \alpha &= \frac{1}{4} + \frac{1}{8}(f_{cra,1} + f_{cra,2}) \\ \rho &= \alpha\rho_a + (1-\alpha)\rho_c \end{aligned} \tag{4}$$

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

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.

If one sensor node is stationary while another sensor node is moving, these nodes 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.

3.4 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 ($T_H = \frac{H}{f_f}$ 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 offline 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 the other parameters are held constant. 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 correlating and non-correlating 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 unrelated experiments is 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 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

Impact of sampling rate. The simulation results presented in Figure 1 show that increasing the sampling rate up to about 24 Hz has a significant positive influence on the accuracy of detecting correlated motion. Beyond 24 Hz, the impact is insignificant as shown in the top plot. The bottom plot shows that the sample frequency has no visible

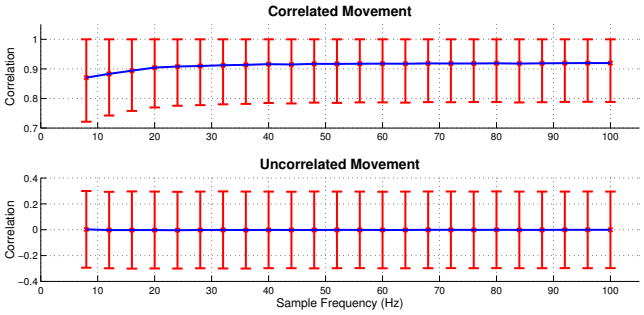

Figure 1. Correlation output statistics at varying sample frequency.

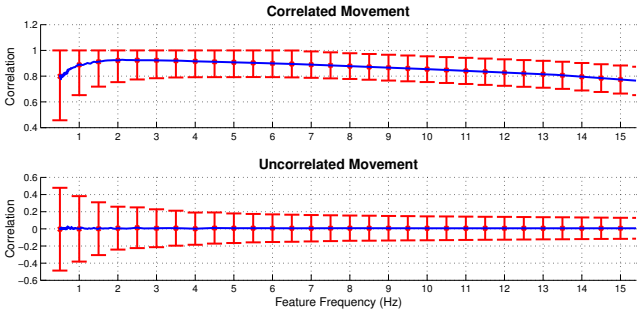

Figure 2. Correlation output statistics at varying feature frequency.

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.

Impact of feature frequency. Figure 2 shows the performance results at varying feature frequencies for both correlated and uncorrelated movement ($f_s = 40$ Hz). 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.

Impact of correlation history. The top plot of Figure 3 shows the accuracy variation with the correlation history length. 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.

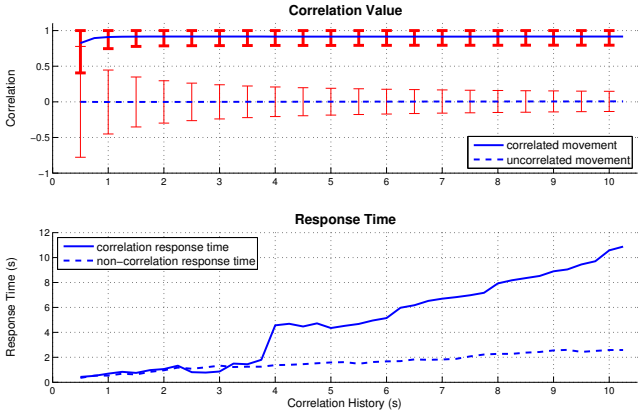

Figure 3. Correlation output and response time statistics at varying correlation history.

The impact of the correlation history length on the response time is shown in the bottom plot of Figure 3. Up to about 3.7 s the response time for correlated and uncorrelated movement are 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.

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. The results are illustrated in Table 1. 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.

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 the next section.

5 Implementation

We use the ProMove [3] wireless inertial sensor nodes for this work. The ProMove board features a 3-D accelerometer and a 3-D digital compass. The main CPU of the sensor node is a low-power MSP430 microcontroller running at 8 MHz. The nodes can communicate wirelessly using a CC2430 SoC, which combines an IEEE 802.15.4-compatible radio with an 8051 CPU. The CC2430 CPU autonomously handles the wireless networking.

The correlation algorithm is implemented on the sensor nodes in a pairwise manner. Both nodes in a pair preprocess the raw signals from their sensors, yielding a two-dimensional feature vector (refer to Section 3.1), which is transmitted from one node to the other. The receiving node performs the correlation calculation. The nodes communicate using the IEEE 802.15.4 protocol. One node acts as the coordinator and broadcasts its feature vector to slave nodes that check their correlation with the coordinator. The sampling and feature extraction tasks running on the slave nodes are coarsely synchronized to the coordinator for proper correlation performance, which is achieved with a precision of one sample.

To investigate the feasibility of our implementation, we ran a benchmark to measure the processing load on the MSP430 processor. The complete implementation uses only approximately 7 % of the processor's time, leaving therefore ample resources available for the high-level application.

6 Tests and Results

To evaluate our implementation, we perform a series of experiments with handling objects equipped with sensors. In each experiment, the user wears one sensor node on a bracelet on his arm and two other sensor nodes are placed on a dumbbell weight and inside a soft foam ball respectively, as shown in Figure 4. 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 gateway 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. Both these issues have a negative impact on the overall performance.

We experiment with four different users, which perform five individual tests. Each individual test lasts for two minutes. The user handles one of the objects with the arm on which he wears the sensor. At random times, a second person uses the objects that are not currently held by the user, thus trying to generate false correlations. Similar to the offline evaluation, the push-buttons on the objects are used for automated annotation of the ground truth.

Thresholds		Response Time (s)		Errors (%)	
corr.	non-corr.	corr.	non-corr.	corr.	non-corr.
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

Table 1. Correlation threshold statistics.

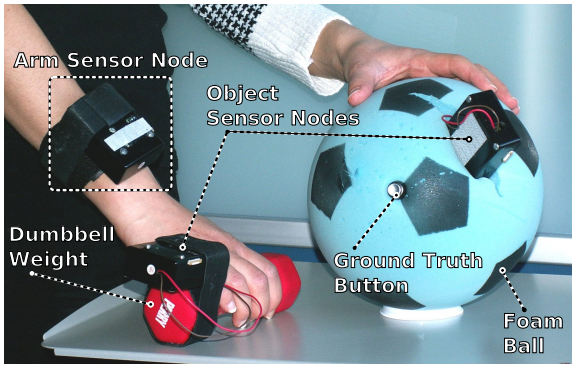

Figure 4. The involved hardware.

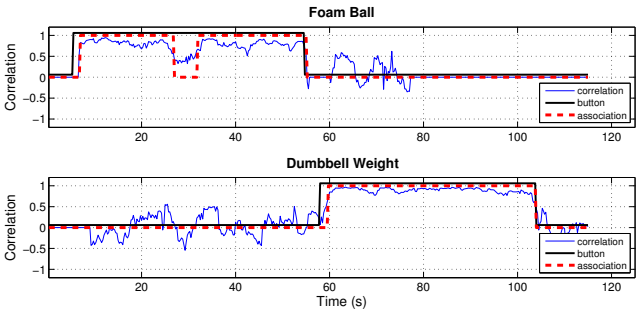

Figure 5. Example of an experiment.

Figure 5 shows an example experiment, comparing the true object use and the output of our detection algorithm. The top plot shows the use of the foam ball while the bottom plot shows the use of the dumbbell weight. At approximately 5 s, the user first picks up the foam ball and moves it for about 50 s. At the same time, the dumbbell weight is moved by another person. Before the 60 s mark, the user lays down the foam ball and takes the dumbbell weight from the other person. He uses the weight until the end of the test, while the other person moves the foam ball until approximately 78 s. As shown in the graph, the algorithm produces the correct result at most instances, with response times lower than 2 s in all cases. A false negative produced by the sensor in the foam ball occurs at 30 s, with the cor-

	Response Time (s)		Errors (%)	
	corr.	non-corr.	corr.	non-corr.
User 1	1.68	1.08	2.23	0.08
User 2	1.53	1.83	1.40	1.17
User 3	2.61	1.02	3.24	0.45
User 4	1.56	1.04	4.66	0.49
Mean Performance	1.84	1.24	2.88	0.55

Table 2. Performance statistics.

relation coefficient dropping below the lower threshold for about 5 s.

Table 2 shows the overall performance of our implementation for all 20 tests. As expected, in most cases the performance is slightly 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 of 3 % of the time and false non-correlation of 0.5 % of the time.

7 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. 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 that handles it. This indicator jumps to another player when the ball is tossed.

Figure 6 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 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 the LEDs on the arm sensors.

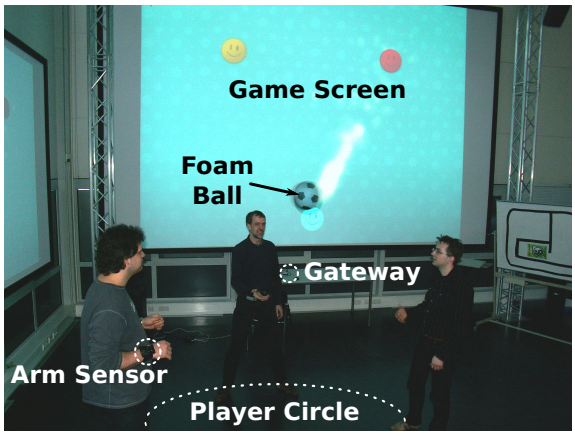

Figure 6. People playing the Hot Potato game.

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 the data logged by the computer running the game indicate an average response time of 2.5 s with a standard deviation of 0.9 s. This is larger than the response time achieved in our 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. Thus even though the response time on the actual sensor nodes remains within the 2 s target, the reaction of the graphical interface is slightly slower.

8 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. In this concluding section, we briefly overview the main advantages, limitations, trade-offs and ideas for future work.

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 *what* 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 can 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 on an energy-efficient wireless communication protocol.

However, calibration factors and magnetic field disturbances can affect the performance of the compass sensor. In most cases, the influence is marginal since the correlation coefficient is not affected by scale or offset differences in the signals. The acceleration feature is not fully compensated for gravity influence in our approach. For movements involving significant rotation, the correlation of the acceleration feature is less reliable and therefore receives a lower weight in our decision logic. Also, with one sensor on the user's wrist and other sensors in the handled objects, there is a possibility that the sensor on the arm is subjected to different motion than the sensor inside the held object. This is due to the wrist joint of the user's arm. In our experiments, however, most movements performed during handling the objects show that both the hand and the lower arm have similar rotation and acceleration intensities, meaning that our algorithm works adequately. Finally, network delays and packet losses can adversely affect the synchronization and lead to a decrease in the correlation performance. The problem is not accumulative, however, meaning that, when the communication quality is restored, the correlation performance gets back to normal.

Our tests and simulations indicate that there is a clear trade-off between the accuracy and the response time of the algorithm. To keep the solution lightweight and feasible for sensor nodes, we reduce as much as possible the sampling rate and feature frequency and we simplify the feature computation, the feature fusion and the decision logic. This has, however, a further negative impact on the achievable accuracy of the algorithm. The response time is further influenced by our effort to minimize power consumption, which aims to minimize the rate of wireless communication. The best trade-off values that we set in the implementation produce an accuracy of 97% for a response time of 2 seconds, while causing a system load on our hardware of about 7 % and exchanging messages at a rate of 4 Hz.

Our detection of object use could be improved by incorporating radio signal strength (RSSI) or other proximity information. This way, nodes that are far apart can be omitted from the correlation process, as these have a low chance of moving together, reducing the number of false positives. Also, to improve the correlation of lateral movement, the acceleration measurements could first be compensated for the influence of gravity by use of a gyroscope. The gyroscope could also supplement or replace the compass as rotation sensor. Additionally, the current prototype sensor nodes are continuously sampling and communicating, which is a significant waste of resources when the nodes are not moving. Using the hardware motion triggers present in most modern digital motion sensors (particularly accelerometers), we aim to mitigate this problem by letting the nodes go to sleep when not moving and waking up once moved. Finally, we would like to generalize our solution towards arbitrary groups of sensors in stead of only a pairwise assessment, making our solution suitable for a much broader set of applications.

References

[1] W. Brunette, C. Hartung, B. Nordstrom, and G. Borriello. Proximity interactions between wireless sensors and their application. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 30–37, San Diego, CA, USA, 2003. ACM.

[2] M. Buettner, R. Prasad, M. Philipose, and D. Wetherall. Recognizing daily activities with RFID-based sensors. In *Proceedings of the 11th international conference on Ubiquitous computing*, pages 51–60, Orlando, Florida, USA, 2009. ACM.

[3] Inertia Technology. *ProMove wireless inertial sensor node*. http://www.inertia-technology.com.

[4] F. Kawsar, K. Fujinami, and T. Nakajima. Exploiting passive advantages of sentient artefacts. In *Ubiquitous Computing Systems*, pages 270–285. 2006.

[5] J. Lester, B. Hannaford, and G. Borriello. Are you with me? - using accelerometers to determine if two devices are carried by the same person. In *Pervasive Computing*, pages 33–50. 2004.

[6] R. Marin-Perianu, M. Marin-Perianu, P. Havinga, and H. Scholten. Movement-Based group awareness with wireless sensor networks. In *Pervasive Computing*, pages 298–315. 2007.

[7] R. Mayrhofer and H. Gellersen. Shake well before use: Authentication based on accelerometer data. In *Pervasive Computing*, pages 144–161. 2007.

[8] D. Patterson, D. Fox, H. Kautz, and M. Philipose. Fine-grained activity recognition by aggregating abstract object usage. In *Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on*, pages 44–51, 2005.

[9] M. Strohbach, G. Kortuem, H. Gellersen, and C. Kray. Using cooperative artefacts as basis for activity recognition. In *Ambient Intelligence*, pages 49–60. 2004.

[10] E. M. Tapia, S. S. Intille, and K. Larson. Activity recognition in the home using simple and ubiquitous sensors. In *Pervasive Computing*, pages 158–175. 2004.

[11] J. Wu, A. Osuntogun, T. Choudhury, M. Philipose, and J. Rehg. A scalable approach to activity recognition based on object use. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, 2007.