Fall-detection Simulator for accelerometers with in-hardware preprocessing

Sebastian Fudickar, Christian Karth, Philipp Mahr, Bettina Schnor
Institute of Computer Science, University of Potsdam
Germany
[fudickar|ckarth|pmahr|schnor]@cs.uni-potsdam.de

ABSTRACT

Mobile fall-detection systems that use accelerometers (as the ADXL 345) with data pre-processing capabilities, enable processors to remain longer in low power modes and therefore can achieve extended device lifetimes. Since fall-detection on these accelerometers is partially executed in hardware, the development and comparison of fall-detection algorithms requires direct evaluation on the hardware and increases complexity. We introduce a fall-detection simulator for the development and comparison of fall-detection algorithms for accelerometers with and without partial in-hardware pre-processing. In addition comprehensive records of fall-situations and daily living activities were generated for the simulator from recording movements. With the help of the simulator, the sensitivity of a given fall-detection algorithm could be improved from 33% to 93%.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Signal processing systems; I.6.7 [**Simulation and Modeling**]: Simulation Support Systems; J.3 [**Life and Medical Sciences**]: Health

General Terms

Algorithms, Experimentation

Keywords

Fall-Detection, Simulator, Accelerometer

1. INTRODUCTION

The demographic age-related changes increase the importance of elderlies health and longer self-defined living. Falls are seen as a major cause for sudden health aggravation of elderly people. The early notification of falls increases the probability of full rehabilitation and the safety of elderlies. If a fall occurred, fall-detection systems imediately can notify care givers, emergency services or beloved ones automatically.

Along these systems, mobile solutions are most promising, because of privacy concerns [14] and lower setup efforts as discussed in Section 2. Such mobile systems must achieve high device lifetimes and high fall-detection precision (defined by minimal false-negative rates so called maximal sensitivity and lower false positive fall rates so called maximal specifity [2]). Mobile fall-detection algorithms extract fall patterns from accelerometer or gyroscope data. If remaining continuously in high-power states, its energy efficiency could be critical for device lifetimes. Novel accelerometers achieve a power consumption below 1 mW and include data pre-processing capabilities. Since continuous data-processing is handled already by the accelerometer, the processor can mainly remain in low-power states. As a result, these accelerometers with in-hardware pre-processing can extend the device lifetimes significantly and are included in an increasing amount of smart phones and special purpose devices. The evaluation of fall-detection algorithms for accelerometers with in hardware pre-processing is not trivial since the parameter settings of the in-hardware parts are essential.

Smart phones most commonly include accelerometers, and therefore may be well suited to integrate fall-detection functionality (e.g. included in Apps). While the devices' operating systems (such as Android or IOs) support hardware independent application portability in general, it becomes more complicated for fall-detection applications. Accelerometers interfaces such as control registers, parameter settings, interrupts, maximal supported sensing frequencies (for the ADXL345 depends on the system bus) and preprocessing functionality vary. In the ADXL345 6 control-registers can be adjusted. In our experiments, the adjustment of a single parameter lead to sensitivity decrease of 29% (for resetting the stable time threshold from 1 to 2 seconds) and 33% (for resetting the free fall threshold 0.75g auf 0.5625g). The hardware functionalities, however, are directly accessed by the fall-detection algorithms with in-hardware preprocessing, to achieve higher efficiency. Furthermore, variances in the accelerometers placement on the circuit board influences the movement characteristics as well and therefore require parameter adjustments per device type. The manufacturer's default parameter settings (with 100% precision) for the ADXL345 lead in our tests to 33% correct detected falls of overall 84 falls. Even the wearing position of devices may influences the falls movement characteristics (e.g. the impact intensity or the duration of each falling stage) and may be covered as well.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PETRA2012 June 6 - 8, 2012, Heraklion, Crete, Greece.
Copyright 2012 ACM 978-1-4503-1300-1/12/06 ...$15.00.

As a result, fall detection applications that take advantage of the in-hardware pre-processing functionality must be ported to each device and accelerometer type. Next to adaptation of the functionality and registers, the fall-detection algorithm portation requires testing on the devices. The testing with real falls seams inpracticle, since it requires the availability of each device and the execution of multiple falls per device. Instead an automated approach is required for developers to test fall-detection algorithms with in-hardware pre-processing on recorded fall-situations.

The comparability of algorithms is critical either, next to the portability to other devices and chips. Fall-detection algorithms for accelerometers with in-hardware pre-processing combine functionality of both the accelerometer (for pre-processing) and the processor (for post processing and parameter adjustments) and thereby have an increased complexity compared to monolithic fall-detection algorithms without such functional distribution. While monolithic algorithms without such functional distribution can be tested on general purpose devices with recorded fall situations. In contrast, testing distributed algorithms is more challenging since the accelerometers must be accessible, to alter parameter settings. As a result distributed algorithms can only be validated by multiple real falls. Since each fall has unique movements (such as acceleration or rotation), the comparison of algorithm's sensitivity and specificity without fall-situation records is meaningless. Therefore, a meaningful comparison of algorithms as well as parameter settings can only be achieved via prerecorded fall situations. The comparability, the development and portability of fall-detection algorithms for accelerometers with hardware pre-processing capabilities therefore needs specific simulators that include real time execution of the hardware functionality, to overcome the necessity of frequent real world testing. This work presents a simulator for accelerometers with hardware pre-processing capabilities that enables testing and comparability of fall-detection algorithms under defined (prerecorded) situations for multiple chip sets and devices (once recorded). The simulator is presented in Section 5.

Available recorded raw records of fall-situation and activities of daily living (ADLs) is presented in Section 6. By comparing two algorithms via the records we show the impact of on-chip threshold-settings on the precision in Section 7. The most sensitive algorithm was tested in the efficient mobile unit (EMU) [11], a prototype hardware platform as well. The EMU (as shown in Figure 1) is designed to assist elderly persons within the KopAL system [12] and includes an ADXL345 accelerometer[1] [9]. The comparison of the simulated and realistic sensitivities do not differ significantly according to Pearson chi^2 test, but indicates potential further extensions, as discussed in Section 7.

2. RELATED WORK

A major difference of existing fall-detection approaches is the device mobility. The sensor devices are either mobile attached to the user [26] or are placed stationary in the user's environment as found in vision based [14, 4] or pressure pad systems [24, 18]. Along these solutions less intrusive mobile systems are most promising, because of privacy concerns [14], lower setup efforts and location-independent usage.

[1] The ADXL345 in the EMU is connected via a I^2C Bus with a maximal data rate of 800 Hz.

Figure 1: The EMU assists elderlies and detects falls with an ADXL345 accelerometer

Such systems exemplisticly, detect of user movements (such as fall patterns) via accelerometers and gyroscopes, integrated in devices that are worn by the user [19]. Based on such movement data, falls can be either detected by pattern recognition or by threshold based systems. Since pattern recognition (such as Kernel Fisher Discriminant [27] or Hidden Markov Models [13]) are too compute intensive for handheld and low cost sensor devices, sensor readings in this case are streamed over wireless channels to external computers for calculation. Both, high processor load and continuous use of wireless interfaces, result in short device lifetimes. In contrast, threshold based algorithms [1] are less calculation intensive and therefore can be executed on mobile devices [15, 23, 8], excluding the necessity to stream. Threshold based algorithms continuously analyze device movements recognized by accelerometers. Based on the characteristic movements during a fall-situation (as described in Section 4), the movements during a fall and before, during and after an impact are identified by passing predefined thresholds. By including orientation changes (as sensed by gyroscopes) the sensitivity and specificity of threshold based algorithms can be increased [19]. However the recognition of orientation changes can be identified on the base of accelerometer readings via the changes of standard gravity of the earth while the device is in stable position [6]. Jia [16] recently introduced a threshold based algorithm, that includes pre-processing on the accelerometer and therefore enables the processor to remain in low-power modes until, a free fall was detected. Fall characteristics change regarding the detection position [1, 5, 17]. For attachment sites of an accelerometer for fall detection, the chest [1] resp. the waist [17] was reported as the best location. But in Bourke et al. [1], no waist sensor was used. In Kangas et al. [17], no sensor was placed in the chest. Different threshold selections and sensor placements were investigated in [5] to make a methodological comparison of fall detection algorithms. The study shows that post-fall posture recruitment leads to lower false alarm ratios for the considerd methods.

3. ENERGY EFFICIENCY

Energy efficiency is necessary for acceptable device lifetimes. In the case of fall-detection the most relevant components are sensors such as accelerometers, gyroscopes or magnetometers for sensing, processor for calculation and eventually wireless interfaces for communication tasks. The

power consumption of these components is discussed within this section.

Current accelerometers consume less than 1 mW when active [25]. Therefore, accelerometers may remain continuously active, with low impact on the overall device lifetimes. Magnetometers have a significantly higher power consumption, e.g. the AKM Semiconductor AK8973 3-axis magnetometers [7] which is included in the IPhone 3GS and the HTC Desire consumes 22mW in active state. Therefore magnetometers should not remain activated continuously. The power consumption of current gyroscopes with around 90 mW [10] is even higher. As a result, gyroscopes should not be activated continuously, but instead are well suited for periodic sensing or should be limited to critical situations.

Next to the power consumption of the sensing components itself, other components that participate during fall-detection (such as processors or wireless interfaces) are relevant as well.

Sorber et. al. emphasize the necessity of processors' deep sleep modes, since "low-power modes were designed to save power while providing interactivity, not to enable always-on functionality" [22]. The processor should remain most of the time in low power (deep-sleep) states and therefore in case of fall-detection should not participate in the continuous processing of data. Since three axial accelerometers such as the ADXL345 include pre-processing functionality of the sampled accelerometer data, the processor can remain in deep-sleep mode most of the time.

Data streaming over wireless transceivers for external calculations is critical and therefore should be used sporadicly to limit the influence of Wi-Fi's [21], Bluetooth's [20] or cell communication's [3] high power consumptions.

As a result, fall-detection algorithms that mainly use accelerometers, especially such with pre-processing data in the accelerometer (instead of been processed by the processor) maximize device lifetimes and therefore are recommended.

4. ADXL345 FALL MOVEMENT CHARACTERISTICS

This section descibes the fall characteristics which apply (mainly) to movements of the hip, where the EMU is worn (as shown in Figure 4). For a tri-axial accelerometer, fall situations are characterized by multiple sequential events, as shown in Figure 2. While the following desciption is applicable to threshold based fall detection algorithms in general, the parameter settings are described acording to Jia [16].

Within a fall situation, the falling body experiences zero-gravity during free falls. The free fall is the initial event of each fall situation and therefore is identified first. For the ADXL345, the gravity must drop below the free fall threshold THRESH_FF (as described in Table 1) for a minimal duration of TIME_FF. Other algorithms (such as [1]) exclude a minimal duration for free fall detection but instead detect them only via threshold value.

Once a free fall occured, the algorithm tries to detect an impact, which is given, if the acceleration values of all three axis exceed the STRIKE_THRESHOLD. The maximal duration between free fall and impact recognition may not exceed the duration defined in STRIKE_WINDOW. If no impact was detected within this period, no fall was detected and the algorithm resets the accelerometer to detect further free falls.

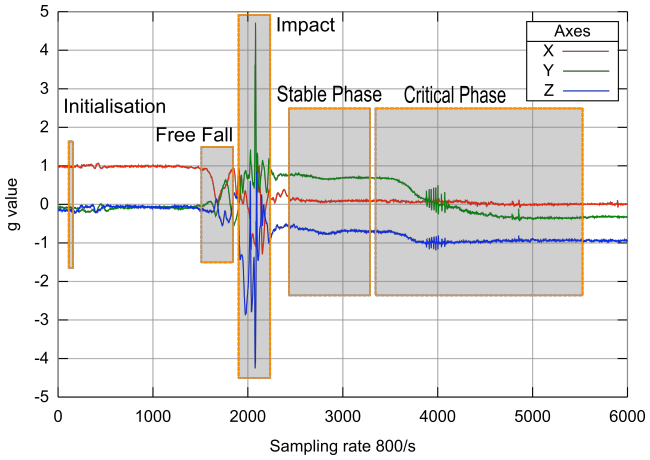

Figure 2: States of a fall shown for a frontal fall without loss of consciousness

In a fall situation the impact is followed by a stable phase. During the stable phase, all axis' acceleration values drop below the NOMOVEMENT_THRESHOLD, for a minimal duration of STABLE_TIME. A fall is recognized, if a free fall, an impact and a stable phase were detected sequentially. Therefore the detection of the stable phase is essential for the fall detection, but does not indicate a loss of consciousness.

Instead, loss of consciousness is detected during the critical phase, which follows the stable phase. If no movements were recognized during the critical phase (for 10 seconds), a loss of consciousness can be assumed, caused by the previous fall.

The above described algorithm requires the processor to adjustments accelerometer parameters (such as thresholds or times) dynamicly.

Since ADLs such as sitting down or stair walking [19] may contain similair movements (free fall phases and impacts) as found during fall situations, threshold based algorithms may cause high false-positive rates. An appropriate threshold adjustment is essential to achieve sufficient specifities of fall detection algorithms.

5. FALL-DETECTION SIMULATOR

To enable automized parameter adjustments and a meaningful comparison of algorithm's sensitivity and specificity, this work introduces a simulator for fall detection algorithms with in-hardware pre-processing. The simulator utilizes pre-recorded fall situations and ADLs to overcome the necessity of frequent physical falls. A physical setup (of eventually multiple hardware systems) is not required to implement multiple algorithms (if fall records for the platforms are available).

As shown in Figure 3, the simulator takes the following input parameters:

- A configuration file defines the simulation environment by including initial accelerometer's register settings and the algorithm parameters (such as data rate, free fall threshold and timing variables).
- The fall-detection algorithm, which should be
- Either fall-situation or ADLs records

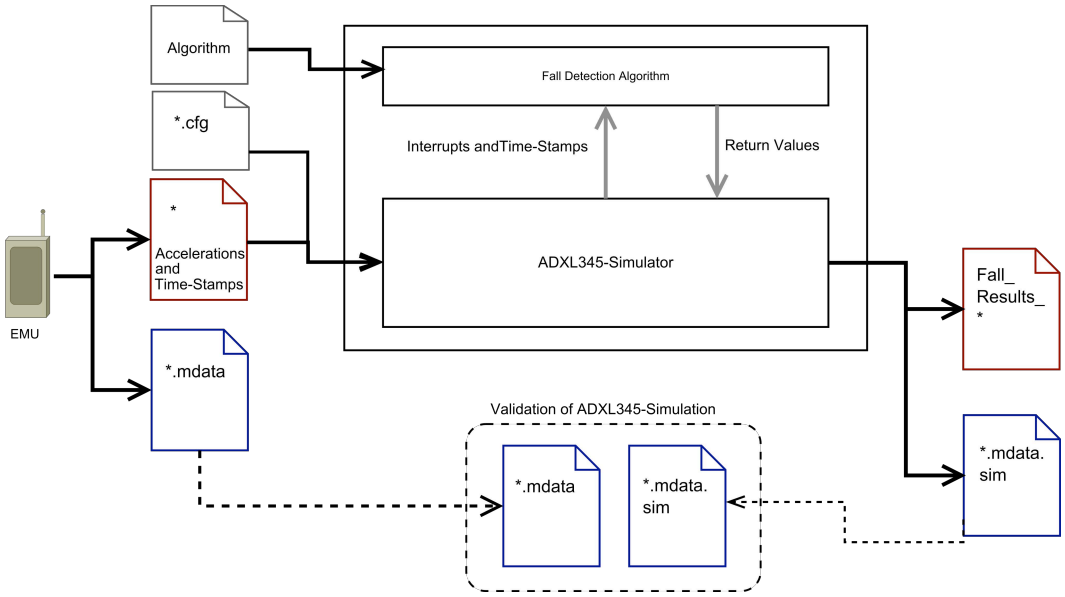

Figure 3: System architecture of the fall-detection simulator

The simulator can be used to test the complete combination of parameter settings for an algorithm automatically by generating multiple configuration files. Next to the initial parameter settings, the accelerometer's execution parameters can be dynamically adjusted by the fall-detection algorithm itself, during a simulation run. Next to the fall-detection algorithm the processing engine includes a representation of the calculations that are executed on the accelerometer (see Figure 3). The accelerometer's internal algorithms has to be adapted for additional accelerometers. During a simulation run, the fall-situation record-parser transmits entries according to the relative recording time to the virtual accelerometer. The ADXL345-simulator interprets these data according to the accelerometer's current configuration and may send virtual interrupts to the fall detection module. Interrupts indicate state changes and may require the algorithm to alter the accelerometer's state and parameter settings (depending on the current state, algorithm and interrupt type).

The simulator generates the following output:

- Fall-detection results contain the loaded configuration and detected fall types, which are enhanced by a timestamp.
- In addition lists of the generated internal interrupts with the timestamp are generated.

Currently, the simulator supports the EMU hardware platform, which contain the ADXL345 accelerometer. The simulator's available pre-processing functionality (namely free fall, activity and inactivity recognition) was validated by a movement set, in which the raw fall data was logged and processed on the accelerometer. Next to the raw data, all interrupts were logged. The generated interrupts of a simulation run were compared with the hardware interrupts regarding time, count and type. These values were identical.

To support additional devices or accelerometers the following steps are necessary. Additional fall-situation records (for devices including supported accelerometers) allow the evaluation of algorithms for alternate devices. To support devices with other accelerometers, the accelerometers specific functionality must be implemented. Since functionalities often are similar between accelerometers required adjustments are propably minor. These potential extensions should be as well validated via the above described approach.

6. FALL-SITUATION AND ADL RECORDS

Accelerometer data records of fall-situations and ADLs are required to simulate the fall-detection for specific devices. Each record consists of a timestamp, and acceleration data of each of the three axes, with maximal available frequency (of 800Hz). Accelerometers must be calibrated for the recordings (and interrupts) to asure data accuracy.

Currently, fall-situation records exist for the ADXL345, worn on a hip belt as shown in Figure 4. These records include 6 falls per type (namely frontal falls, backward falls, falls to the left and to the right, falling after the stand up, falling from the bend down - while picking up a book and falling from stairs as suggested by [26]) each with and without loss of consciousness. Images of recorded falls (as shown in Figure 5 for a frontal fall) are as well included to enable an interpretation of the fall characteristics and for the recording of comparable measurement-sets with other devices. The overall 84 fall situations, each with a duration of 15 seconds, were conducted by three probands between 20 and 30 years old [2], including both genders.

Next to the fall-situations, ADLs were recorded to evaluate false-positive rates. These ADLs include 10 sitting sequences for various types of seats (such as chair, stools and sofas) and 5 minutes of daily routines such as flower watering and stair climbing from the three probands. All ADLs were executed with multiple speeds, to cover potential influence of speed on the false-positive rates [19].

[2] Seniors were excluded from the study, to prevent injuries during falls.

Register Settings	Description	Optimized algorithm	Original algorithm
THRESH_FF	free-fall threshold	0.75 *g* (0x0C)	0.75 *g* (0x0C)
TIME _FF	minimal free-fall duration	30 *ms* (0x06)	30 *ms* (0x06)
STRIKE_THRESHOLD	minimal impact per axis	2 *g* (0x20)	2 *g* (0x20)
STRIKE_WINDOW	maximal delay of impact	500 *ms* (0x19)	200 *ms* (0x0A)
STABLE_TIME	minimal stable-phase duration	1 *s* (0x01)	2 *s* (0x02)
NOMOVEMENT _THRESHOLD	maximal acceleration during stable-phase	0.4375 *g* (0x07)	0.1875 *g* (0x03)

Table 1: Register settings of original [16] and optimized fall-detection algorithm

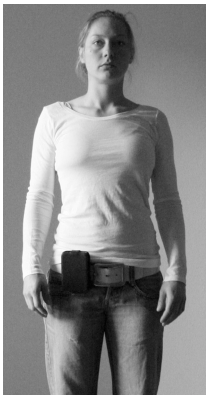

Figure 4: Device placement for recording of fall-situation and ADLs

Figure 5: Sequence of a fall to the left

Since the EMU is operated with a Linux distribution, all records were collected on the EMU itself via a kernel module [3].The accelerometers internal FIFO and auto-sleep mode were deactivated during the recording [9]. The (filtered) raw data measurement range was set to 16g with a resolution of 13 bit and a scaling factor of 3.9 mg/LSB (least significant bit).

The use of such records are only meaningful for the specific accelerometer they are recorded with. Under different conditions (including calibration, scalability factors, system architecture, bus type and transmission speed) the results are not representative for other devices. Since even factors such as device form-factors, accelerometer placement on the PCB or the bus utilization by other peripherals may affect the transmission quality and sensor readings, we suggest the collection of records for each device type. The fall-situation and ADLs recordings and the simulator are available for download [4].

[3] See http://lxr.free-electrons.com/source/drivers/input/misc/adxl34x.c?a=blackfin

[4] Further details are available under http://www.cs.uni-potsdam.de/bs/research/al/

7. OPTIMIZATION AND COMPARISON OF ALGORITHMS

We evaluated two threshold based fall-detection algorithms regarding their sensitivity and specificity. Herefore the fall situation and the ALDs records of the EMU were used. Both algorithms detect falls via the accelerometer and utilize in-hardware pre-processing capabilities. Next to the basic (software) algorithm [16], which was designed for the ADXL345 by the manufacturer, we evaluated an optimized version of it.

The optimized version of the algorithm differed from the original version by the following adaptations:

- Initial values for recognition of orientation values are dynamically adjusted, while in the original version they were staticly set. Orientation changes are recognized if the difference between the current and recent orientations exceeds a threshold of 0.7g.
- Optimal parameter settings for the accelerometer in the EMU were determined with the simulator. This evaluation covers the Cartesian product of 6 parameters, each with up to three different configurations as shown in Table 1. Out of 96 potential parameter combinations the most sensitive parameter setting was identified. Compared to the original parameter settings, the STRIKE_WINDOW period was extended, the STABLE_TIME was halfed and the NOMOVEMENT _THRESHOLD was increased.

Table 1 shows the optimal parameter settings for the EMU in comparison to the original parameter settings which were reported by the manufacturer to detect 100% of the falls. Our simulation of the original algorithm with the suggested parameter settings indicated a detection quota of only 33%. Since the manufacturer evaluated the algorithm on different hardware the different results may be caused by the characteristics of the devices and the accelerometers lower detection precision (which is caused by the potential use of a different bus as discussed in Section 5). In addition, the output data was previously filtered (by the accelerometer) which might create mathematical inprecision, potentially affecting the detection.

The extended fall-detection algorithm with the optimal parameters achieved a higher sensitivity of 93%, as shown in Table 2.

In addition, the fall-detection precision of a Linux kernel module implementation of the extended algorithm with the most precise parameter set (S1) was evaluated on the EMU (worn on a hip belt) by two persons. All falls corresponded to the schema described in Section 6 (excluding of the stair fall). Therefore, the algorithm was evaluated with 48 falls, from which 41 falls were detected correctly.

Fall Motion	Falls opt.	Falls orig.
6x Falling Forward	6	0
6x Falling Forward (l.o.c.)	6	3
6x Falling Backward	6	3
6x Falling Backward (l.o.c.)	6	2
6x Falling to the Left	6	1
6x Falling to the Left (l.o.c.)	6	3
6x Falling to the Right	4	2
6x Falling to the Right (l.o.c.)	5	2
6x Falling after the Stand up	5	1
6x Falling after the Stand up (l.o.c.)	6	2
6x Falling from the Bend down	6	2
6x Falling from the Bend down (l.o.c.)	6	2
6x Falling from the Stairs	5	3
6x Falling from the Stairs (l.o.c.)	5	2
84	78	28

Table 2: Fall-detection sensitivity of original algorithm (orig.) and optimized algorithm (opt.) including loss of consciousness(l.o.c.)

The implementation achieved a sensitivity of 85% (and 83% sensitivity for critical falls with loss of consciousness (20 of 24). According to the Pearsons Chi^2 test, the sensitivity of the simulation and the real falls had no significant difference (p=0.164). The remaining differences between the sensitivity of the simulation and real tests can result from in hardware measurement value adjustments and by the inaccuracy of the hardware algorithm.

8. SUMMARY

This work introduces a simulator for accelerometer based fall-detection algorithms which utilize the hardware support of novel chips such as the ADXL345. The simulator can help developers to compare the sensitivity of different algorithms for a given device. In addition, developers may efficiently develop and optimize fall-detection algorithms by automated evaluation on pre-recorded real falls. With the EMU worn on a hip belt 96 falls of three probands for multiple fall-types with and without loss of consciousness, were recorded. In addition, ADLs were recorded from three probands, covering activities such as sitting and stair climbing with different speeds. These records of falls and ADLs can be used for the sensitivity and specific evaluation of fall-detection algorithms within the simulator. By automated simulation of the cross product of all potential parameter settings for an extended fall-detection algorithm we illustrated the efficiency of the simulator during development and optimization of novel fall-detection algorithms. A sensitivity increase of 60% was recognized by comparing the fall-detection sensitivity of the optimized fall-detection algorithm (93%) with the original one. In addition, a sensitivity of 85% was achieved when testing a corresponding Linux kernel module implementation of the optimized algorithm on the EMU with 48 real falls. Therefore, the presented simulator offers a novel method to compare the sensitivity of multiple fall-detection algorithms for specific accelerometers, while helping to reduce the necessity of real falls.

9. ACKNOWLEDGMENTS

We like to thank Max Frohberg and Sebastian Taube for their assistance regarding hardware and OS and the daring probands of the fall recordings C. Karth, M. Müller and F. Hämmerling for their stunning falls.

10. REFERENCES

[1] G. A.K.Bourke, J.V.O'Brien. Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm. *Gait & Posture*, 26:194–199, 1 July 2007.

[2] D. G. Altman and J. M. Bland. Statistics notes: Diagnostic tests 1: sensitivity and specificity. *BMJ*, 308(6943):1552, 6 1994.

[3] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM on Internet measurement conference*, IMC '09, pages 280–293, New York, NY, USA, 2009. ACM.

[4] G. Bieber, A. Hoffmeyer, E. Gutzeit, C. Peter, and B. Urban. Activity monitoring by fusion of optical and mechanical tracking technologies for user behavior analysis. In *Proceedings of the 2nd International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '09, pages 45:1–45:6, New York, NY, USA, 2009. ACM.

[5] P.-K. Chao, H.-L. Chan, F.-T. Tang, Y.-C. Chen, and M.-K. Wong. A comparison of automatic fall detection by the cross-product and magnitude of tri-axial acceleration. *Physiological Measurement*, 30:1027––1037, 2009.

[6] J. Chen, K. Kwong, D. Chang, J. Luk, and R. Bajcsy. Wearable sensors for reliable fall detection. *Conf Proc IEEE Eng Med Biol Soc*, 4:3551–4, 2005.

[7] A. K. M. Co. Ak8973 3-axis electronic compass. Asahi Kasei Microsystems Co., 2007. `http://www.asahi-kasei.co.jp/akm/en/product/ak8963/ak8963.html`; accessed March 1, 2012.

[8] J. Dai, X. Bai, Z. Yang, Z. Shen, and D. Xuan. Mobile phone-based pervasive fall detection. *Personal Ubiquitous Comput.*, 14:633–643, October 2010.

[9] A. Devices. Digital accelerometer adxl345 (rev. c). Analog Devices. `http://www.analog.com/static/imported-files/data\_sheets/ADXL345.pdf`; accessed March 1, 2012.

[10] A. Devices. Programmable low power gyroscope adis16250 rev(d). Analog Devices, 2009. `http://www.analog.com/static/imported-files/data_sheets/ADIS16250_16255.pdf`; accessed March 1, 2012.

[11] S. Fudickar, M. Frohberg, P. Mahr, and B. Schnor. Energy efficient wireless communication for indoor location systems. In *In preparation for GreenCom 2012*, 2012.

[12] S. Fudickar, B. Schnor, J. Felber, F. J. Neyer, M. Lenz, and M. Stede. An orientation system for patients with dementia. In *Behaviour Monitoring and Interpretation Book of Well Being*, 2011.

[13] R. K. Ganti, P. Jayachandran, T. F. Abdelzaher, and J. A. Stankovic. Satire: a software architecture for smart attire. In *Proceedings of the 4th international conference on Mobile systems, applications and*

services, MobiSys '06, pages 110–123, New York, NY, USA, 2006. ACM.

[14] R. Hartmann, F. A. Machot, P. Mahr, and C. Bobda. Camera-based system for tracking and position estimation of humans. In *DASIP*, pages 62–67. IEEE, 2010.

[15] C.-N. Huang, C.-Y. Chiang, J.-S. Chang, Y.-C. Chou, Y.-X. Hong, S. J. Hsu, W.-C. Chu, and C.-T. Chan. Location-aware fall detection system for medical care quality improvement. *Multimedia and Ubiquitous Engineering, International Conference on*, 0:477–480, 2009.

[16] A. D. N. Jia. An-1023 - fall detection application by using 3-axis accelerometer adxl345. Analog Devices. `http://www.analog.com/static/imported-files/application_notes/AN-1023.pdf`; accessed March 1, 2012.

[17] M. Kangas, A. Konttila, P. Lindgren, I. Winblad, and T. Jamsa. Comparison of low-complexity fall detection algorithms for body attached accelerometers. *Gait & posture*, 28:285–91, 2008.

[18] P. Leusmann, C. Mollering, L. Klack, K. Kasugai, M. Ziefle, and B. Rumpe. Your floor knows where you are: Sensing and acquisition of movement data. *Mobile Data Management, IEEE International Conference on*, 2:61–66, 2011.

[19] Q. Li, J. A. Stankovic, M. A. Hanson, A. T. Barth, J. Lach, and G. Zhou. Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information. *Wearable and Implantable Body Sensor Networks, International Workshop on*, 0:138–143, 2009.

[20] J. Paek, J. Kim, and R. Govindan. Energy-efficient rate-adaptive gps-based positioning for smartphones. In S. Banerjee, S. Keshav, and A. Wolman, editors, *MobiSys*, pages 299–314. ACM, 2010.

[21] E. Shih, P. Bahl, and M. J. Sinclair. Wake on wireless: an event driven energy saving strategy for battery operated devices. In *Proceedings of the 8th annual international conference on Mobile computing and networking*, MobiCom '02, pages 160–171, New York, NY, USA, 2002. ACM.

[22] J. Sorber, N. Banerjee, M. D. Corner, and S. Rollins. Turducken: hierarchical power management for mobile devices. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, MobiSys '05, pages 261–274, New York, NY, USA, 2005. ACM.

[23] F. Sposaro and G. Tyson. iFall: an Android application for fall monitoring and response. *Conference proceedings: Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2009:6119–6122, 2009.

[24] P. Srinivasan, D. Birchfield, G. Qian, and A. Kidane. Design of a pressure sensitive floor for multimodal sensing. In *Proceedings of the Ninth International Conference on Information Visualisation*, pages 41–46, Washington, DC, USA, 2005. IEEE Computer Society.

[25] STMicroelectronics. Lis302dl mems motion sensor 3-axis - $\pm 2g / \pm 8g$ smart digital output '"piccolo'" accelerometer. STMicroelectronics, 2007. `http://www.kako.com/neta/2008-017/lis302dl.pdf`; accessed March 1, 2012.

[26] C. Wang, C. Chiang, C. Huang, and C. Chan. Development of a fall detecting system for the elderly residents. *The Second International Conference of Bioinformatics and Biomedical Engineering*, 0:1359–1362, 2008.

[27] T. Zhang, J. Wang, P. Liu, and J. Hou. Fall detection by embedding an accelerometer in cellphone and using kfd algorithm. *Journal of Computer Science*, 6(10):277–284, 2006.