Inexpensive and Automatic Calibration for Acceleration Sensors

Albert Krohn, Michael Beigl, Christian Decker, Uwe Kochendörfer, Philip Robinson and Tobias Zimmer
Telecooperation Office (TecO)
Universität Karlsruhe

***Abstract*— In this paper, we present two methods for calibration of acceleration sensors that are inexpensive, in-situ, require minimum user interaction and are targeted to a broad set of acceleration sensor applications and devices. We overcome the necessity of orthogonal axes alignment by extending existing calibration methods with a non-orthogonal axes model. Our non-orthogonal method can furthermore be used to enable automatic calibration for 1- or 2-axes accelerometers or realize a simultaneous mass-calibration of sensors with minimum effort. The influence of noise to the presented calibration methods is analysed.**

I. Introduction

Calibration is an important issue in sensor based systems as it is the only way to ensure a predictable quality of delivered information. Traditionally, calibration is done in a process during production time. The process is very costly and sometimes requires manual steps which makes it difficult to lower costs at this point. Recalibration is necessary after some time for most sensor systems and can often only be done at particular sites and hence incurs additional high costs. Ubiquitous computing environments differ in many aspects from traditional sensing settings. Ubiquitous Computing devices are used as peripheral devices disappearing from the user's awareness. Therefore it can be said that the majority of Ubiquitous Computing devices will be factually and metaphorically invisible [1]. As a consequence, users tend to ignore intense device administration, even though operational conditions may be extremely rigourous, such that errors due to neglected recalibration accumulate. Furthermore, the circumstances of device deployment in Ubiquitous Computing environments typically sees them embedded into other objects, hindering access and opportunity to perform mechanical recalibration. The amount of computational devices deployed is also envisioned to be on a higher scale in comparison to traditional sensor systems. We refer to hundreds of devices spread in the environment, comparable to the random distribution of low costs, everyday objects and consumer electronics to date. This introduces new challenges for management and maintenance especially for the calibration and recalibration processes.

In research, this development has already started with the instrumentation of everyday objects such as cups [2]. In order to allow embedding, the devices must be very small, priced in the cent range and absolutely maintenance free. Costly calibration including high priced hardware is not an option for these applications. Calibration has to be done in-situ without user intervention and additionally has to be done under the assumption that the sensor technology is cheap. A widely used sensor technology in Ubicomp is the MEMS-type acceleration sensor. The sensors are small and quite accurate. Various platforms and applications that incorporates them have been developed. Examples are Lancaster's DIY Smart-Its [3], TecO's particle computers [4],the WearPen and TiltPad [5]. Additionally, companies like Crossbow [6] and Silicon Designs [7] offer products for easy integration of accelerometers in other products even for 3-axial acceleration measurements.

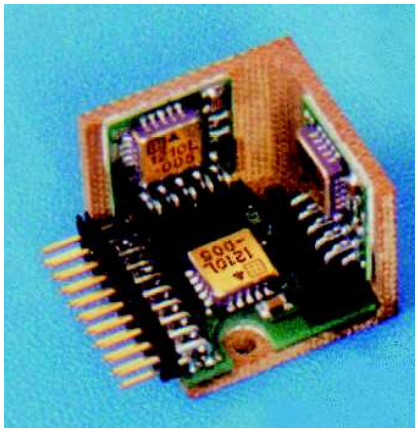

Fig. 1. 3-Axes accelerometer built out of 1-axis sensors © Silicon Designs [7]

However, calibration is necessary, especially if acceleration sensors are used collaboratively. For example, if a 2-axes acceleration sensor device and a second 1-or 2-axes acceleration sensor device are used together in one system - e.g. attached to the same object - both devices can spontaneously form a 3-axes acceleration sensor device. Many of the mentioned platforms build their 3-axes acceleration sensors out of two 2-axes acceleration sensors that are orthogonally mounted. One example is the mentioned design from Silicon Designs. Figure 1 [7] shows how three 1-axis sensors are mechanically combined to a three-dimensional acceleration sensor.

In this paper we present a method that allows us to calibrate 3-axes accelerometers simply and simultaneously. It also makes it possible to combine several 1- or 2-axes accelerometers and to form a cheap 3-axes acceleration sensor on the fly and then calibrate the underlying 1- or 2-axes sensors. Such calibration can be done in parallel for many

combined 3-axes sensors:

Hundreds of these sensor devices can be calibrated simultaneously by just putting them into one box and calibrating them all with only some measurements in different orientations of the box. During the measurements, the 1- and 2-dimensional sensors would build 3-axes accelerometers by virtually combining them to 3D sensor systems. The main problem to deal with in these setting is that the axes of the combined sensors are not orthogonal but randomly oriented in the box. This problem is not restricted to simple and cheap sensors it is also a problem for most off-the-shelf 3 dimensional acceleration sensors. The axes of the sensors are often not perfectly aligned to 90° due to mechanical impreciseness. The model that is presented in section IV presents a solution for such 3D sensor calibration of sensors that have non-orthogonal axes either if they are already assembled or are assembled on-the-fly to form a 3-axes acceleration sensor.

One approach for in-situ calibration of *3-axes* acceleration sensors involving the parameters *offset* and *scale* was introduced by Lukowicz et al. in [8]. They proposed a method for the calibration that needs only some random measurements in different orientations taking advantage of the earth's gravity field.

Our paper focuses on in-situ calibration methods especially regarding noise in the measurements and tilted (non-orthogonal) accelerometer systems. We will extend the current approaches in order to take care not only of offset and scale within the calibration, but also of the *orthogonality* of the axes. The characteristic of in-situ calibration will be kept.

The paper proceeds with a short overview of calibration methods involving only offset and scale. Section III investigates the influence of measurement noise and tilted axes on the methods. Section IV deals with non-orthogonal axes and gives an extension of the traditional approaches. Implementation consideration are covered in section V before we conclude in section VI.

II. State of the Art Calibration

In many applications there is a need for a calibrated acceleration measurement, but many sensors - especially MEMS types - are not calibrated after production. Instead the sensor have a sensitivity s and an offset o on each of their axes which lead to measurements that do not represent the actual physical value. Before presenting the two most important calibration techniques, we define some expression used throughout the paper:

- the physical acceleration is named as $\vec{x} = (x, y, z)^T$. It is measured in multiples of the earth's gravity g
- the measurement offsets of the axes of sensors are named o_x, o_y, o_z and the scalings s_x, s_y, s_z, whereas a perfect system would have $\vec{o} = \vec{0}$ and $\vec{s} = \vec{1}$
- the values measured by an uncalibrated, *orthogonal* 3D acceleration sensor are named $\vec{u} = (u, v, w)^T$; uncalibrated means here that offset and scaling errors are still present in the measurements
- the values measured by a *calibrated, non-orthogonal* 3D acceleration sensor are named $\vec{r} = (r, s, t)^T$; calibrated means here that offset and scaling errors have been discovered and the measurements are corrected accordingly
- the interrelationship between measurements and real physical values in an orthogonal system is for x-axis: $x = (u - o_x)/s_x$;. For the y- and z-axis: the according equations.

A. Rotational Calibration

The method is described in [9] and determines the offset and the scale factor for each axis separately. Hereby, an axis (e.g. the x-axis) of the acceleration sensor is oriented to the earth's gravity centre and kept stationary. It is exposed to 1g and rotated and exposed to -1g. The measured values (in g) in both positions are $u_{max,x}$ and $u_{min,x}$. Solving the equation system

$$1 = \frac{u_{max,x} - o_x}{s_x}$$
$$-1 = \frac{u_{min,x} - o_x}{s_x}$$

will result in the offset o_x and scale factor s_x for this axis:

$$o_x = \frac{u_{max,x} + u_{min,x}}{2} \quad (1)$$
$$s_x = \frac{u_{max,x} - u_{min,x}}{2} \quad (2)$$

In order to find u_{max} and u_{min} the rotation has to be carried out very slowly to minimize the effect of dynamic acceleration components. The accuracy of the method relies significantly on the accuracy of the alignment.

B. Automatic Calibration

Another method for calibration of 3-axes accelerometers has been presented in [8]. It does not require a certain series of pre-defined positions like the method explained above and is therefore very practical in mobile settings. It can perform a calibration in-situ after a complete device has been assembled. It is also suitable for fast mass-calibration as the sensor virtually calibrates itself. The idea is to use the earth's gravity force as a known static acceleration on a 3-axes accelerometer when a sensor has no dynamic component applied on it. Is this state of being stationary detected the following equation (see [8], page 2) is valid:

$$|\vec{x}| = \sqrt{(x^2 + y^2 + z^2)} = 1 \quad (3)$$

With the according offsets and scale errors of the accelerometers, equation (3) extends to:

$$((u - o_x)/s_x)^2 + ((v - o_y)/s_y)^2 + \\ +((w - o_z)/s_z)^2) = 1 \quad (4)$$

With equation (4) the six unknown calibration variables $(o_x, o_y, o_z, s_x, s_y, s_z)$ can be solved when creating a six lined equation system using six earth's gravity vectors measured in different orientations of the sensor. The six necessary measurements should be significantly different from each

other to guarantee a stable convergence of the non-linear solver. Additionally, it must be assured, that all three axes are *perfectly aligned to be orthogonal*, otherwise the precondition for the algorithm is violated and the results will be errorneous.

C. Concerns Using the Presented Calibration Methods

When calibrating acceleration sensors in MEMS technology we generally experience the problem of noise introduced in the measurement. Noise can have different reasons like thermal noise, quantization noise or noise introduced by A/D conversion. Noise during the calibration corrupts the measurements and therefore results in imprecise calibration. It is necessary to have a quantitative insight of the influence of noise to a calibration process.

Further, the calibration of the axes' angles is *not* addressed in either calibration method. A complete calibration of a 3D accelerometer includes as well the calibration of the axes' angles to 90° to be able to calculate the physical acceleration $\vec{x}$ from the measured value $\vec{u}$. Some settings (mentioned in section I) show that it's necessary to provide calibration methods for tilted axes. Furthermore, the axes' angles calibration is of special interest for the automatic calibration method as the orthogonality is the precondition for the validity of equation (3). With titled axes, the automatic calibration method will not work!

III. Influence of Noise

It is impossible to avoid errors on the calibration due to noise during the calibration process. To figure out what negative influence on the calibration results must be expected from a known noise level, we simulated different noise levels with an even distribution and a maximum of 5, 10, 20 and 50 mg and applied them on virtual measurements before using them in the two calibration procedures. Figures 2 and 3

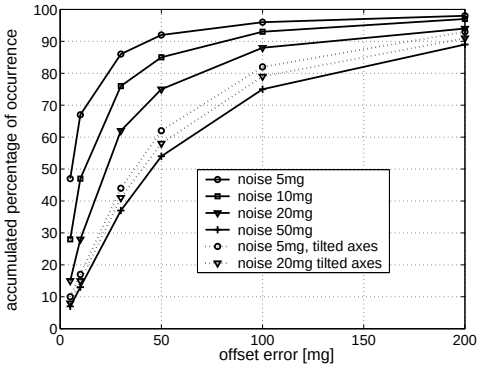

Fig. 2. Offset errors for simulated automatic calibration

show the accumulative error distribution of the calibration parameter offset using the two presented calibration methods. The offset errors are shown in absolute error, the scaling errors in Figures 4 and 5 relatively. Reading out from Figure 2 with parameter $noise = 20mg$ at abscissa $offseterror = 50mg$ gives the value 75. That means that statistically for 75% of the calibrations that have a 20mg noise level, the offset error

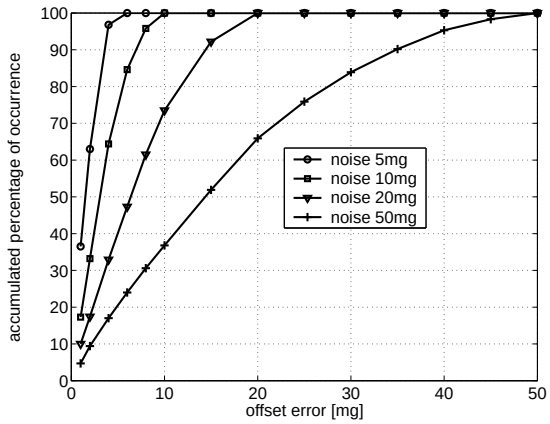

Fig. 3. Offsets errrors for simulated rotational calibration

in the results of the calibration will be less or equal 50mg. The 75% can be regarded as the statistical trust level of the offset error being less or equal 50mg in the calibration when applying 20mg of noise on the measurement.

A quick example should show how the Figures 2 and 3 for the offset errors can be compared to Figures 4 and 5 for the scaling errors: An error of 10% in scaling would cause an error in a measurement of approx. 100mg when stationary pointing towards ground. Therefore, as a rule of thumb, 1% error in scaling can be compared to 10mg in offset error. The graphics are useful when a certain target accuracy is desired. Then, one can read out what the (mean) necessary input accuracy should be. For example, a desired target accuracy of 50mg in offset and 5% in scale with a trust level of 95% would require a noise level around 5mg using the automatic calibration.

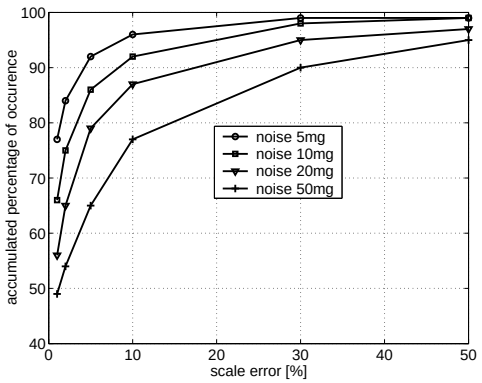

Fig. 4. Scaling errors for simulated automatic calibration

When comparing e.g. the 10mg noise curves of Figure 2 and 3, one can see that the 95% confidence level for rotational method gives an offset error of around 100mg where the rotational method reaches 8mg! On the first sight, the rotational method would be the clear winner. But the result needs to be interpreted in a broader context. The solver of the non-linear equation array causes additional errors on the automatic calibration. And the noise model on the rotational

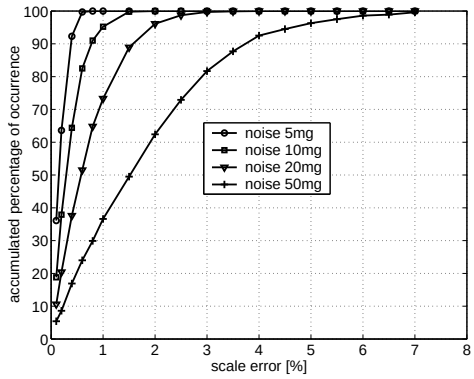

Fig. 5. Scaling errors for simulated rotational calibration

algorithm does *not* include the errors that occur when the sensor was not measured in it's minimum and maximum but slightly off to these points. In [8] the authors gain a significant improvement in accuracy by averaging a series of results of one sensor.

IV. Non-Orthogonal Axes

As already mentioned, the calibration of axes is not supported by either presented calibration method. For the rotational calibration, the axes are calibrated independently and the offset and scaling calibration is not negatively affected by angular displacement. For this reason, we propose an extension of the rotational algorithm in section IV-B to calibrate possible axes displacements as a second step after the standard offset and scaling calibration.

For the automatic calibration method, it is a necessary precondition that the axes are orthogonal. We simulated some cases, where the angle was off by 5°. These curves can be found as dotted lines in Figure 2. The error is significantly worse than without angular displacement, which motivates the extension of the algorithm to be able to deal with non-orthogonal systems.

A. Model of the Tilted Accelerometer Axes

The idea of the extension of the known equations and algorithms to be used with non-orthogonal angles reflects the wish to have three perfectly aligned accelerometers in a bundle. As this is not always the case, a mathematical conversion can simulate a perfectly aligned sensor array by calculating the *virtual orthogonal* values from the measured and *tilted* values. For this conversion, the off-axis angles of the acceleration sensor must be known. Figure 6 shows a euclidean coordinate space $\{x, y, z\}$ representing an ideal acceleration sensor array. For simplicity and w.l.o.g. we assume that the offset and scaling errors are not present and therefore only model the direct relationship between the tilted measurements $\vec{r}$ and the real physical acceleration $\vec{x}$. Later, we can include the offset and scaling errors to formulate an expression including all negative influences. A vector $\vec{a}$ would be exactly measured with its three components x, y and z. But as the axes are not orthogonal, the values measured by the acceleration sensors (r, s, t) do not represent the correct 3D acceleration applied on the sensor.

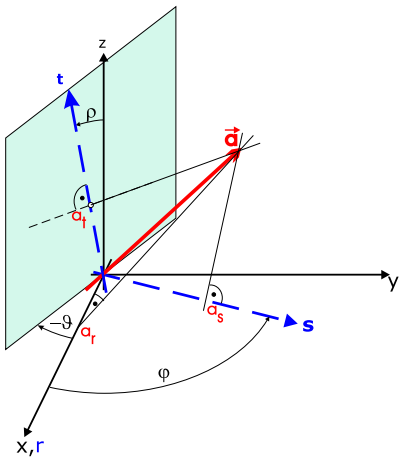

Fig. 6. Tilted Axes of accelerometers

The axes of the accelerometers have three levels of freedom named as φ, ρ and ϑ. For simplicity, we assume a sensible setting where the three accelerometer axes are close to the desired euclidean axes x, y and z and span an R^3 vector space. The axis of the accelerometers are named r, s and t. W.l.o.g, the r-Axis will be defined equal to the x-Axis. Then secondly, the s-Axis will lie in the x-y-plane and therefore only have one degree of freedom: the angle between r-Axis ($x - Axis$) and s-Axis named φ. The tilted t-Axis has two degrees of freedom: ρ and ϑ. The ϑ is the angle between the positive x-Axis and a virtual plane that contains the z-Axis and the t-Axis. The ρ defines the angle between t- and z-Axis. It is defined in positive x direction if $\vartheta = 0$.

For the model, the first necessary formulas are the unit vectors of the new r, s- and t-axes. They are (in the euclidean system):

$$\vec{e_r} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \qquad \vec{e_s} = \begin{pmatrix} \cos\varphi \\ \sin\varphi \\ 0 \end{pmatrix},$$
$$\vec{e_t} = \begin{pmatrix} \cos\vartheta \cdot \sin\rho \\ \sin\vartheta \cdot \sin\rho \\ \cos\vartheta \end{pmatrix} \tag{5}$$

Generally, the measurement that a one-axis acceleration sensor derives from an arbitrary acceleration vector is the acceleration vector fraction, which is parallel to the acceleration sensor axis. Therefore, an arbitrary acceleration sensor like the $\vec{a}$ in figure 6 needs to be projected in an orthogonal manner on the r, s- and t-axis in order to find out what the according sensors would measure. The measurements are named a_r, a_s, a_t. Notice that this projection is different from a coordinate space transformation where the vector $\vec{a}$ would be decomposed in the directions parallel to the three axis. The projection from $\vec{a}$ on the s-axis is not parallel to the $r - t$-plane but orthogonal to the s-axis!

Therefore, the next step is the projection of $\vec{a}$ on the three axes r, s and t. The auxiliary plane

$$E : \vec{e_s} \circ (\vec{x} - \vec{a}) = 0$$

is intersected with the auxiliary even

$$g : \vec{x} = \lambda \vec{e_s}$$

resulting in

$$\lambda = \vec{e_s} \cdot \vec{a}.$$

This defines the orthogonal projection vector of $\vec{a}$ on the s-axis being

$$(\vec{e_s} \cdot \vec{a}) \cdot \vec{e_s}$$

and therefore its length being the measurement of an acceleration sensor pointing in $\vec{e_s}$-direction measuring an arbitrary vector $\vec{a} = (x, y, z)^T$. The length of this vector

$$\begin{aligned} s = \quad & |(\vec{e_s} \cdot \vec{a}) \cdot \vec{e_s}| = \vec{e_s} \cdot \vec{a} = \\ v = \quad & x \cos\varphi + a \sin\varphi \end{aligned} \tag{6}$$

Doing so equally for the w-axis results is

$$\begin{aligned} t = \quad & |(\vec{e_t} \cdot \vec{a}) \cdot \vec{e_t}| = \vec{e_t} \cdot \vec{a} = \\ = \quad & x \cos\vartheta \cdot \sin\rho + y \sin\vartheta \cdot \sin\rho + z \cos\rho \end{aligned} \tag{7}$$

The solution for the r-axis is trivial:

$$r = \quad x \tag{8}$$

The three expressions (6),(7),(8) define the measurements of the acceleration sensor carrying the tilt angles φ, ϑ and ρ. If a measurement is gathered with such a sensor, the virtual values in an ideal orthogonal system could be calculated by inversing the expression for r, s and t. This results in:

$$x = \quad r \tag{9}$$

$$y = \quad \frac{s - r\cos\varphi}{\sin\varphi} \tag{10}$$

$$z = \quad \frac{t}{\cos\rho} - r\cos\vartheta\tan\rho - (s - r\cos\varphi)\frac{\sin\vartheta}{\sin\varphi}\tan\rho \tag{11}$$

Now, an interrelationship between the tilt angles of the axes φ, ϑ, ρ, the measurements $(r, s, t)^T$ and the real physical acceleration $(x, y, z)^T$ has been found.

B. Using the Non-Orthogonal Axes Model

The results from the previous section can generally be used in two ways: Firstly, they enable the automatic calibration model to be used in settings where the axes are not orthogonal. To do so, the expressions (9), (10), (11) can simply be put in place of the measurements $\vec{u}$ and in equation (4) to solve for the - now nine - unknown variables (three angles, three offsets, three scalings). It is therefore possible to extend the automatic calibration method to systems with non-orthogonal axes. Doing so, the simplified model of (9), (10) and (11) without scaling and offset errors is extended to a model including both the tilted axes and the offset and scaling errors.

Secondly, the results can be used to extend the rotational calibration to a complete calibration including the axes' angles. The rotational calibration method itself is not vulnerable to non-orthogonal axes systems as it calibrates the axes independently through the use of the minimal and maximal values. But it does not provide an easy way to also calibrate a possible axes angle displacement. For this reason, we extend the rotational calibration method after its completion (the offset and scaling are calibrated) with three steps to calibrate a possible axes angular displacement. For the following equations, the vector $\vec{r}$ is - like above - the measurement vector *including* the correction in scale and offset but *not including* the angular correction. The angle calibration process is as follows:

1. position the 3D sensor that the measurements $r = s = 0$. As the r, s, x and y-axis all lie in the same plane (see figure 6), the real physical values are $x = y = 0, z = g$. With (7):

$$t = z\cos\rho \Leftrightarrow \rho = \arccos(\frac{t}{g}) \tag{12}$$

2. position the 3D sensor that the r-measurement is maximized. Then the real acceleration $\vec{x} = (g, 0, 0)^T$, because x and r have been defined to be parallel. In (6):

$$r = g\cos\varphi \Leftrightarrow \varphi = \arccos(\frac{r}{g}) \tag{13}$$

3. in the same position, but using equation (7) we get:

$$t = g\cos\vartheta\sin\rho \Leftrightarrow \vartheta = \arccos(\frac{t}{\sin\rho}) \tag{14}$$

With these simple calculations, the three tilt angles can be calculated using only one extra position for the sensor (the position with u being maximized is anyway necessary for the offset and scaling calibration). It is important to notice that the angle calibration for the rotational calibration has to take place after the calibration for offset and scaling has been finished as we need both $\vec{x}$ and $\vec{r}$ for the calibration of the angles.

V. Implementation Considerations

Both previously presented calibration approaches are investigated for implementation on our particle computer platform [4]. On the sensor boards we are using two ADXL210 accelerometers mounted orthogonal to each other for a 3D accelerometer. Usually this mounting is done manually, which denies an accurate positioning of the sensors to each other. In the discussion we assume that data from the ADXL is provided in the gravity unit g. Sampling the raw data from the ADXL and converting them in g values can be done very quickly and with a manageable complexity [9].

A. Rotational Calibration

The complexity of this method can be directly derived from the equations (1) and (2). For each axis one has to compute scale and offset by just one addition and one division operation. Latter can be replaced by a right shift by one. The angles φ, ρ and ϑ are computed by applying the arccosine function.

B. Automatic Calibration

This calibration method requires solving a non-linear equation system with nine unknown variables. Standard mathematical literature proposes the Newton's method for this problem:

$$F'(\underline{x}_n)\underline{\delta} = -F(\underline{x}_n),\ \underline{x}_{n+1} = \underline{x}_n + \underline{\delta} \tag{15}$$

Hereby, $F(\underline{x}_n)$ represents all nine equations written as a (9x1) vector, $\underline{x}_n = (o_x, o_y, o_z, s_x, s_y, s_z, \vartheta, \varphi, \rho)^T$ is the calibration's solution vector and $F'(\underline{x}_n)$ is the (9x9) Jacobi matrix of all partial derivations of $F(\underline{x}_n)$. The method works in 2 steps, which are iterated to produce better approximations in each step:

1) solve the linear equation system (LES) (15) for an auxiliary variable δ
2) compute next iteration solution by $\underline{x}_{n+1} = \underline{x}_n + \underline{\delta}$

In the first iteration an appropriate vector $\underline{x}_0$ as start value has to be selected. Further, nine measurements of 3-axis acceleration values $(m_x, m_y, m_z)^T$ gathered from nine different positions are neccessary for the computation of $F(\underline{x}_n)$ and $F'(\underline{x}_n)$. The following operations are necessary per iteration:

1) For compilation of LES: 121 additions, 163 multiplications, 80 divisions, and 7 trigonometric operations. The Gauss algorithm solving this LES needs additional 292 additions and 321 multiplication operations.
2) For $\underline{x}_{n+1}$ computation: 9 addition operations

Newton's method implies further constrains one has to take care of. If using an inappropriate start value, then Newton's method can get stuck in an infinite loop without producing improved approximations. One solution is a solver variation called damped Newton's method. Hereby, an additional contraction factor is introduced. As a result, the contraction speed of the method will slow down. Math literature publishes strategies for selecting this factor optimal. However, the damped Newton's method adds additional complexity for computing this factor. In order to keep the overall computing effort low, the number of iterations should be kept low. However, to achieve a reasonable accuracy the stop criterion has to be carefully selected. Usually, the computation is aborted, if $|x_{n+1} - x_n| \leq \epsilon(\epsilon > 0)$, where x_n is the approximated null after n steps, ϵ is the accuracy requirement, and $|\ |$ is the euclidian distance. A reasonable result is often achieved in less then 10 iterations if the method's convergence order is 2. However, the damped Newton's method will converge only with linear speed implying more iterations.

C. PC vs. Microcontroller

Currently, we have implemented the automatic calibration method only on a PC receiving the measured acceleration sensor values from the particle computer sensor board (Figure 7).

The PC implementation of the automatic calibration cooperates with the particle computer. Latter delivers the measurements for the Newton's method as g values from all three axes. Thereby the microcontroller already ensures that all measurements are taken in stationary situations. The values

Fig. 7. Particle sensor board

are given in $[mg]$ units. The challenge for the implementation on the PIC18f6720 microcontroller of the particle computer is the effort needed to achieve accuracy and speed. Although addition and multiplication are done on this processor in hardware, the internal registers only support 8bit integer operation. But, for automatic calibration floating point operation is mandatory. This implies more effort for each mathematical operation. Furthermore, this also adds a certain extra memory usage. In our experience the code size for arbitrary arithmetic functions increases significantly when going from integer to floating point or even including trigonometric functions. Table I summarizes this observation with an arbitrary small algorithm. All numbers are in bytes.

	Integer	Float
ROM	234	1292
RAM	13	50

TABLE I
MEMORY USAGE OF MATH FUNCTIONS [IN BYTES]

As an alternative to the more complex, unsupervised in-situ calibration we implemented a supervised calibration method that is able to run on small microcomputer systems. The rotational calibration we used here is significantly less complex than the automatic calibration and therefore consumes less memory and computing power resources. Using this method we do not rely anymore on the availability of a PC based backend program that was used to do the computational part of the calibration process in the automatic calibration method. We implemented the rotational calibration with a very convenient user interface: the user needs only to roll the sensor board over each axes (x,y,z) until the respective LED is switching off. He is not requested to perform the movement in each of the axes in a certain way, so also inexperienced users can perform the calibration task. We studied the procedure with three users so far. They where shown how to handle the sensor boards and were then requested to calibrate the sensor systems about 10 times without help. All of the users were able to handle the task and all calibrations were done successfully. The achieved accuracy with this method was extremely precise and the calibrated sensors delivered values with deviation from the actual physical values that are in the order of the noise level of the hardware.

VI. Conclusion and Future Work

In this paper, we presented two methods for the calibration of acceleration sensors with minimum user interaction. We extended the traditional methods with the capability to work with *non-orthogonal* axes to meet more practical requirements in real world settings especially for Ubiquitous and Pervasive Computing. We implemented and tested both methods. The methods are now able to handle settings with cheap sensors that have non-orthogonal axes and do not need any (expensive) additional equipment or complicated procedure for calibration. While the automatic calibration runs completely unsupervised but requires some computing resources the rotational calibration is capable to run on small 8 bit microprocessors with limited resources. Both methods allow to calibrate cheap acceleration sensors in a convenient way without any additional equipment. They are also able to combine three 1-dimensional or two 2-dimensional acceleration sensors to a 3-dimensional sensor on the fly. Another possible use of the methods is mass-calibration of hundreds of sensors, as the calibration can be done in parallel. We have shown throughout the paper - backed up by practical experiments - that the results of the calibration is very good (in the order of the noise level of the hardware). Nevertheless, we are continuing our work to eliminate the remaining error sources. We identified that the solver for the non-linear equation is a major source for the errors in the calibration results and will look into customized solutions for this problem in the future.

VII. Acknowledgements

The work presented in this paper was partially funded by the European Community through the project *CoBIs* (Collaborative Business Items) under contract no. 4270 and by the Ministry of Economic Affairs of the Netherlands through the project *Smart Surroundings* under contract no. 03060.

References

[1] Mark Weiser. The computer of the 21st century. *Scientific American*, 265(3):94–104, September 1991.

[2] Michael Beigl, Hans-Werner Gellersen, and Albrecht Schmidt. Mediacups: Experience with design and use of computer-augmented everyday objects. *Computer Networks, Special Issue on Pervasive Computing*, 35(4):401–409, March 2001.

[3] *http://ubicomp.lancs.ac.uk/smart-its/*, accessed: 4/2004.

[4] *http://particle.teco.edu*, accessed: 4/2004.

[5] A. D. Cheok, K. G. Kumar, and S. Prince. Micro-accelerometer based hardware interfaces for wearable computer mixed reality applications. In *Sixth International Symposium on Wearable Computers (ISWC)*, Seattle, Washington, USA, 2002.

[6] *Crossbow. http://www.xbow.com/*, accessed: 4/2004.

[7] *Silicon Designs, Inc. http://www.silicondesigns.com/*, accessed: 4/2004.

[8] Paul Lukowicz, Holger Junker, and Gerhard Tröster. Automatic calibration of body worn acceleration sensors. In *Proceedings of the second international Pervasive Computing conference*, Vienna, Austria, 2004.

[9] H. Weinberg. Using the adxl202 duty cycle output. iMEMS Technologies/Applications AN-604, Analog Devices, 1998.