Open access • Proceedings Article • DOI:10.1109/ISWC.2000.888468

What shall we teach our pants — Source link

K. Van Laerhoven, Ozan Cakmakci

Institutions: Starlab

Published on: 18 Oct 2000 - International Symposium on Wearable Computers

Topics: Wearable computer

What Shall We Teach Our Pants?

Kristof Van Laerhoven and Ozan Cakmakci
Starlab Research
Boulevard St.-Michel 47
B1040 Brussels, Belgium

kristof@starlab.net, ozan@starlab.net

Abstract

If a wearable device can register what the wearer is currently doing, it can anticipate and adjust its behavior to avoid redundant interaction with the user. However, the relevance and properties of the activities that should be recognized depend on both the application and the user. This requires an adaptive recognition of the activities where the user, instead of the designer, can teach the device what he/she is doing. As a case study we connected a pair of pants with accelerometers to a laptop to interpret the raw sensor data. Using a combination of machine learning techniques such as Kohonen maps and probabilistic models, we build a system that is able to learn activities while requiring minimal user attention. This approach to context awareness is more universal since it requires no a priori knowledge about the contexts or the user.

1. Introduction

Making devices aware of the activity of the user fits into the bigger framework of context awareness, which also aims at awareness of environment and the state of the device itself. Existing mobile devices such as mobile phones and personal digital assistants (PDA's) already indicate a growing need for this technology. Since these are not aware of the current situation, they disturb the user (and the environment) in *every* context.

Using an array of hardware sensors to improve applications is not very new. Changing the intensity of displays according to the value of a light sensor is already present in a lot of appliances, for example [15]. A bigger challenge can be found in fusing the data from multiple sensors and mapping it to a high level context-description. Research at Philips [4] and MERL [5] are examples of this approach.

Some interesting work has also been done specifically on activity-recognition: Ashbrook [1] looked at the accelerometer readings from the Twiddler one-handed keyboard to detect walking, Paradiso and Hu [9] worked on pressure sensors in a pair of dancing shoes to control music synthesizers and graphics in a real-time performance. The Acceleration Sensing Glove (ASG) [10] project is another example of activity recognition of hand gestures based on accelerometer information.

Context is a very broad notion, however, this makes it hard to define. Researchers have defined the word 'context' by example, with respect to their own research. Dey and Abowd provided a survey of context and context-aware applications with handheld and ubiquitous computing requirements in mind: "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves" [2].

A lot of activities and contexts are quite personal and can be very different from one person to another ("in the kitchen" or "running very hard", for instance). To overcome these constraints, the system's recognition must become adaptive, so that the user can teach context descriptions to a device in his or her own words. The Technology for Enabling Awareness (TEA) project at Starlab investigates machine learning algorithms [13] to make adaptive context awareness possible, and its results formed the basis for the system used in this paper.

2. General System Overview

In general, to get as much information on a context as possible, it is necessary to use a large amount of different sensors. Sensors used in our experiments include accelerometers (for X-and Y-axis), passive infrared sensors, a carbon monoxide (CO) sensor, microphones, pressure sensors, temperature sensors, touch-sensors and light-sensors (see *Figure 1* for a graph depicting the behavior of various sensors during the contexts: "inside light off"; "inside light on", "inside moving", "outside"

and "outside moving"). This section will discuss a system that was designed to attain context awareness in general,

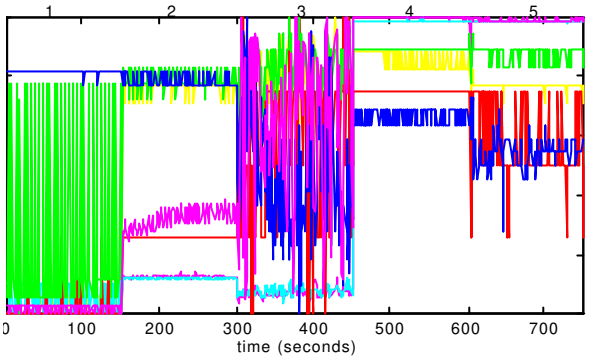

Figure 1. Sensor timeseries during 5 simple contexts.

while section 3 will describe a specific example using 2 sensors.

2.1. Preprocessing the sensor data: cues

If just the raw sensor values would be used as inputs for the machine learning algorithms, performance of the whole system would fall short. However, for some sensors it is possible and even beneficial to use small pre-processing routines to enhance the quality of future clustering. Instead of just looking at the brightness of the light, it is also possible to look at its frequency, which results in easier distinguishing of several types of artificial light. Other sensors like microphones and infrared sensors have similar mini-transformations from the raw sensor data to one or more values that are usually called cues or features in literature (see [11]). In the case of the accelerometers, we have experimented with simple pre-processing routines like standard deviation, the derivatives, and Fast Fourier Transformation (FFT).

One of the conclusions from the introduction was that adaptivity is indispensable in context awareness. As a consequence, machine learning algorithms have to be used. The next sections will discuss the algorithms that are applied to reach adaptive context awareness.

2.2. Self-Organization

The Kohonen Self-Organizing Map (SOM) [6] has a principle that is similar to the self-organization of neuronal functions in the brain: simple units ('neurons') are recruited topologically for tasks depending on the sensory input. The SOM is also known to handle noisy data relatively well, which makes it a sensible choice for clustering the inputs.

At this stage, no explicit teaching or feedback is required from the user. The KSOM clustering algorithm orders the inputs by assigning map-units to each kind of input, and after a while, the resulting map is topologically ordered, i.e. similar inputs activate neighboring units. This "feature map" is often used to visualize high-dimensional data on a two-dimensional display.

The Kohonen SOM is based on earlier work of Willshaw and von der Malsburg [14], where basic units 'compete' for a particular kind of input (hence the name competitive network). For every input that is presented to the map, one unit is selected to be the winner and can adapt itself a bit more towards this input. The adaptation is done by adjusting an internal weight vector (or codebook vector, or prototype vector) $\mathbf{w}$ towards the input vector $\mathbf{x}$:

$$w_{i+1} = w_i + \alpha(w_i - x_i), \alpha \in [0,1]$$

where α is the called the learning rate of the network and i is the number of the training sample.

The rule to find the winner fairly straightforward. The (usually Euclidean) distance is calculated between the input vector and every unit's weight vector, the unit with the closest weight vector wins.

Kohonen introduced the *topologically related* units, so not only the winning neuron gets to adapt its weight vector, but the units that are located in the neighborhood of the winner too. This requires the units to be organized in a certain structure: usually a two-dimensional grid is used (see *Figure 2*), but other unit arrangements or dimensions (like 3D grids or 2D hexagonal) are possible. The update rule thus becomes for *every* unit (or neuron) in the map:

$$w_{i+1} = w_i + \alpha\eta(w_i - x_i)$$

with α again the learning rate between 0 and 1, and η a neighborhood function.

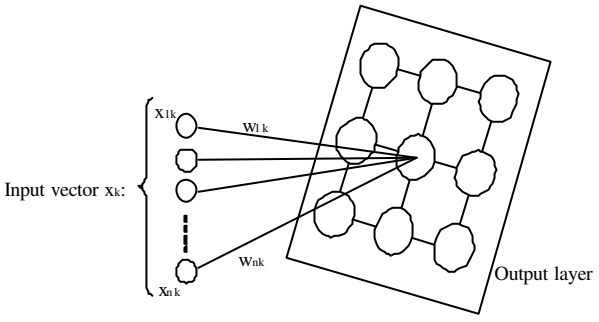

Figure 2. The Kohonen Self-Organizing Map.

After a few iterations, the neurons start to organize themselves in a structured, topological way: different sensor inputs activate different neurons. The activity per neuron can be monitored and afterwards be plotted into a landscape (the x and y axes represent the coordinates on the 2D SOM, and the z axis represents how many times a particular neuron has won – see *Figure 3* for an example). This visualization might be another advantage for the

KSOM, since the lack of insight in the behavior is an often-heard complaint about neural networks. The last activity could be marked on the landscape, so when a user

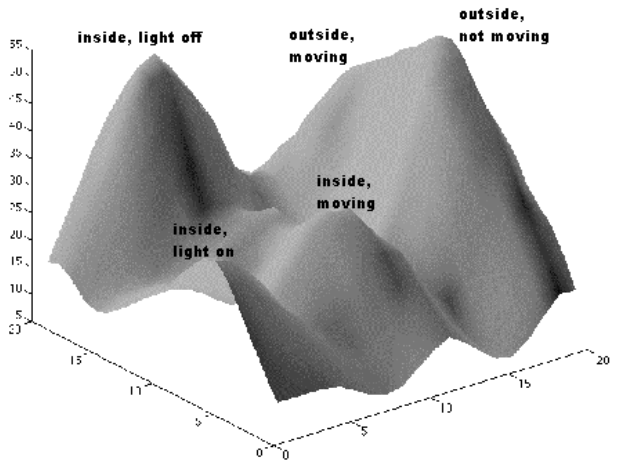

Figure 3. Activity landscape plot of the SOM for sensors-values from five simple contexts.

goes from one context to another, the mark will go from one hill to another one.

The Kohonen Self-Organizing Map has disadvantages, however. The traditional algorithm starts out highly adaptive - with a large learning rate and huge neighborhood radius - and gradually becomes fixed. After this stage, it is hardly capable of learning any more, which poses an obstacle if the system needs to remain adaptive. If the algorithm stays flexible, overwriting might occur of previously stored prototypes. This trade-off is known in the field of machine learning as the *Stability-Plasticity Dilemma* or *Catastrophic Forgetting* [5].

Another problem is the fact that learning slows down as the number of inputs increases: the *curse of dimensionality*. In previous experiments, we used a hierarchy of Kohonen maps (see Figure 4), but this leads to more design issues and is still not a solution when hundreds of inputs are used. See Mitchell [8] for an explanation of both machine-learning problems.

2.3. Classification and Supervision

The result of the clustering layer is a mere identifier that is linked to a label provided by the user. Since one of the system-requirements dictates that user feedback can be given on an irregular basis, it is possible that this label has not been given yet. In that case, a distance-weighted K Nearest Neighbor (KNN) algorithm is responsible for searching the (topologically ordered) cluster map: A voting among the K closest labels on the map, multiplied by their distance to the unlabeled unit, results in the most probable label.

The system that has been described so far is completely based on hardware sensors that can give very noisy signals. That is why a supervision layer is favored. This layer is primarily intended to supervise transitions from one (known) context to another. It uses a probabilistic finite state machine architecture where each context is represented by a state, and transitions are represented by edges between states.

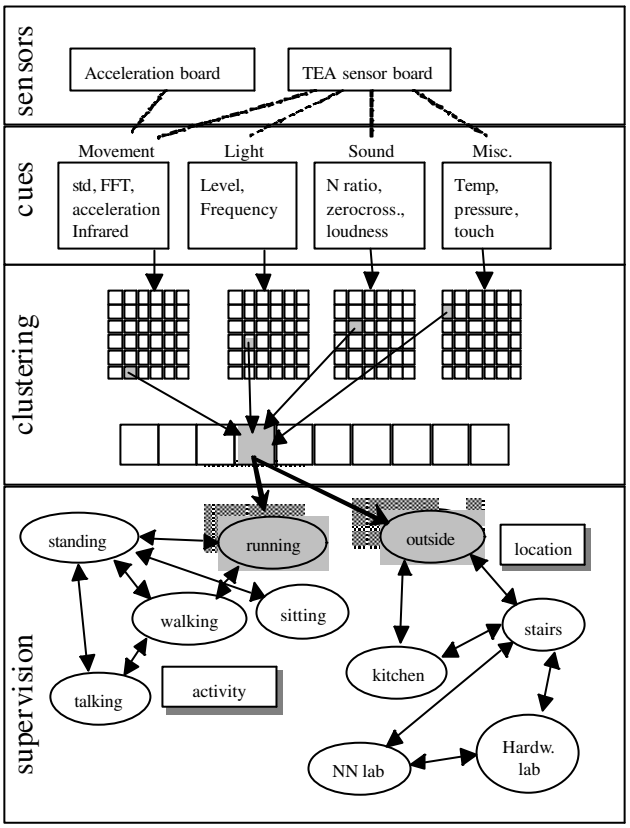

Figure 4. Diagram of the system, starting at the top (sensor-acquisition), going through pre-processing and clustering (using one or more self-organizing maps), and ending at the bottom layer, describing the context by activity and location.

The symbolic model keeps a probability measure for each change of contexts, so every time a transition occurs, the supervision model can check if this really is likely. If a transition is not really probable, the next state is not entered yet, but a buffer mechanism is initiated so that it does become more likely after several tries in a row. More concrete, the transition is done if

$$P(x \rightarrow y) > \frac{(k - \mu)}{k}$$

where $P(x \rightarrow y)$ is the probability that the transition from state x to state y occurs, k is the size of the buffer and μ is

a counter between 0 and k. Each transition to a state is thus dependent on the previous state alone, which makes this model a Markov chain. Every state also keeps track of how much time was spent in a particular context, which controls the flexibility of the SOMs: the newer a context, the more flexible and adaptive the map should be. This method also prevents spikes in the sensor-signals from confusing the whole system and giving a faulty output.

The result is that after some time this model generates a directed graph depicting the behavior of a user with relation to the contexts visited. When the user tends to go from context A to context B rather than to context C, then this will be reflected in the graph's connection strengths. A schematic of the entire system (for both activity and location awareness) is depicted in Figure 4.

3. The Experiment

We chose to implement hardware for capturing *activity*-related data for several reasons. It requires first of all more processing than most location-awareness implementations, which often rely on well-developed hardware such as GPS or beacons. We, on the other hand, tried to minimize the hardware, and focus especially on the software, sensor positioning and user-interaction issues. We believe that activity contributes highly to context awareness and is as important as location, which is also stressed by others [12]. The combination of accelerometers and a pair of pants therefore results in a very attractive experiment.

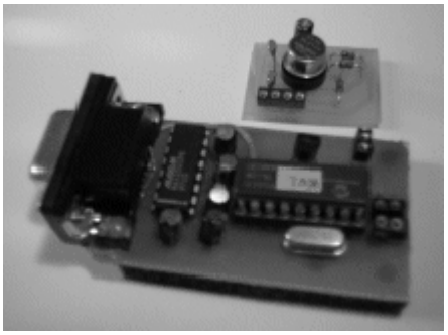

Figure 5. The hardware: a board with the PIC, and a small sensor board with the accelerometer.

Accelerometers are sensors that measure both position and acceleration in one or more directions, depending on the type. In our case, we used two ADXL05s from Analog Devices (one for the X-axis, and one for the Y-axis), which were set to be as sensitive as possible. The values from these devices are read by a PIC microprocessor (PIC16C71 by Microchip), which digitizes them and sends them to a laptop via the serial port. *Figure 5* depicts the two classes of hardware boards that we made for the experiment. By using only two sensors, we can also demonstrate our sensor fusion techniques in the most basic and effective way.

In order to ease sensor placement testing, the accelerometers were fixed on a strap, which could be placed everywhere around the legs, arms, and even the waist. After having made a small list of basic activities we do every day (such as sitting, walking, jumping and running[1]), we observed the sensor-readings and came to the conclusion that the outside of the upper-leg, just above the knee would be the best position for the sensors. After the optimal position is found, the sensors could also be integrated in a pair of pants, by doing the connections with conductive threads. This makes it possible to wash the pants: Only the two small sensor module boards have to be detached, while the power -and data wires can be washed with the pants.

The software on the laptop was running under the Windows operating system and was written in C++. The parameters of the system are listed in *Table 1*.

Table 1. System parameters in the experiment.

Layer	Parameters
Cues Layer	mean(max(50) + std(50)) (10), zero-crossings(50), mean(std(20)) (100).
Clustering Layer	2-dimensional Kohonen map of 20 by 20 neurons with decreasing learning rate (starting from 0.05) and neighborhood radius (5).
Classification Layer	Distance-weighted K-Nearest Neighbors search with K=5
Supervision Layer	Markov Chain

The raw sensor data was processed and mapped to a context description in real-time, and was stored in a datafile as well to enable off-line analysis, comparison and reproduction of the results afterwards. The choice of the set of basic activity contexts we wanted to recognize was based on (1) movement, since it was already established that we would use accelerometers, and (2) classes of movement one could do during the day (while wearing the device). The list we came up with included walking, sitting, running, jumping, climbing stairs, descending stairs and riding a bicycle.

The first design choice would be defining the pre-processing routines on the raw sensor data in the cue layer. The top plot of *Figure 6* shows the data coming

[1] Lying down was not chosen as a basic activity, which might explain the different sensor-placement in the research by Philips (belt worn) [4].

from one accelerometer while sitting, standing up, and walking. The sum of the maximum and the standard deviation over 50 samples (notation: max(50)+std(50)) gave a formula to easily distinguish the three activities. However, to also distinguish other activities, and to keep the system as flexible as possible, other cues must be used, and combined. *Table 1* contains other cues that were used.

Since these algorithms are computationally efficient, implementation on an inexpensive microprocessor (such as Microchip's PIC) is feasible. For a lightweight implementation, simple thresholding could then be sufficient. This is too limited for a general system, though.

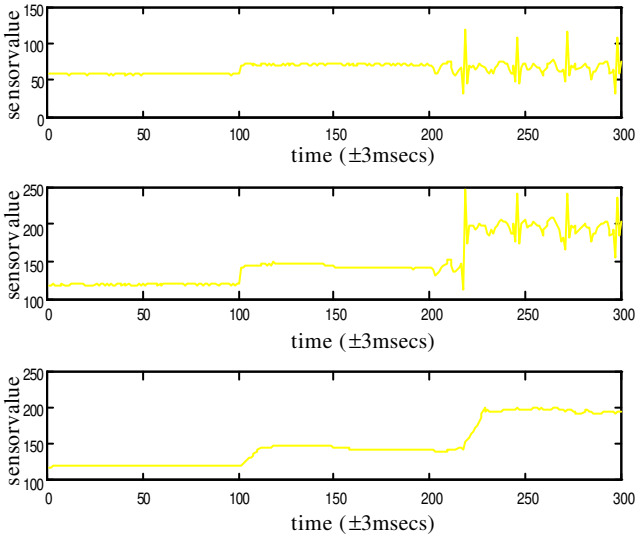

Figure 6. Example of a cue: Plots depicting the sensor values (top), and the sum of maximum and standard deviation (middle) while sitting, standing and walking (each taking 100 samples). The mean of that (bottom) is sufficient for distinguishing all 3 activities.

Although our general system uses a hierarchy of Kohonen maps, we used only one map in this experiment, since the number of inputs is limited enough. The results of the experiments are listed in *Table 2*. The activities were done five times in a row, with each activity taking about a minute. Pressing a designated key during the second repetition began the actual training, and everything was re-trained in the fourth cycle.

Table 2. Experiment results. Cycles 2 and 4 were used for training, 3 and 5 for testing. Cycle one was for pre-ordering the Kohonen SOM.

Activity	Recognition per Cycle	
	3	5
Sitting	96%	96%
Standing	94%	94%
Walking	75%	75%
Running	74%	78%
Climbing stairs	45%	42%
Descending stairs	48%	64%
Riding bicycle	89%	91%

The graph in *Figure 7* shows the resulting activity landscape plot of the Kohonen Map after the experiment. The main activity bubbles represent sitting, standing, bicycling, walking, running, and jumping. The walking bubble includes also "climbing the stairs" and "descending the stairs" (and also a bit of "running"). This can be found back in the results table, where the recognition of these activities are less successful. Better, more specific cues will probably be more beneficial in this case.

Table 2 shows the *average* results of three experiments. Results could already be improved by starting from a pre-prepared Kohonen SOM, instead of one filled with small random values. This would also be safer to avoid overwriting (as explained in section 2 with the stability-plasticity dilemma). However, these results are already promising, since it takes a second or two before the system changes to the appropriate activity description.

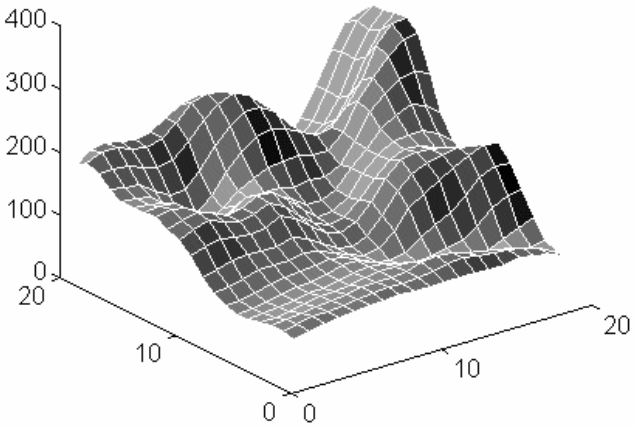

Figure 7. The resulting activity landscape plot.

Finally, the Markov Chain showed indeed some user-behavior towards the activities. The link from "sitting" to "standing", for instance, became very heavy, while the other links were weak or even non-existent. The experimentation period took only several minutes, however, and was done while following a script. The only way to truly assess this part of the system is by doing extended user-testing, which is not very feasible with the prototype system we have so far.

This experiment also shows that the user decides what activities are learned at what time. The interaction is very minimal: the user has to press a button, which is assigned to a description of the activity. This description is just a

string that will be linked to a context identifier, and can be given and modified by the user. A typical scenario looks like this:

- Start walking.
- Press the 'walking' button while walking.
- Stop walking.
- Press the 'standing' button while standing.
- Sit down.
- Press the 'sitting' button while sitting.
- …

Next time the user stands, sits, or starts walking, the system recognizes the activity. Generally, the longer the system is trained, the better the recognition works, but in this simple experiment recognition is already near-100% after the first training for some contexts (as can be seen in *Table 2*). Training new activities afterwards (or retraining old ones) is possible as well, the user needs just to put in the description and press the training button.

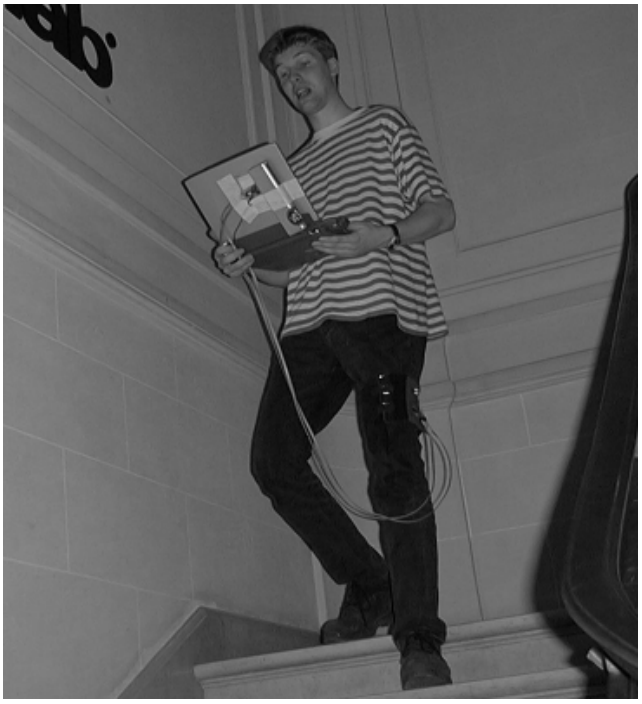

Figure 8. The experiment in progress with the accelerometers placed just above the knee.

4. Future Work

It is clear that improving the sensors-hardware would enhance the entire system. Adding other sensors, like microphones, makes it possible to increase the number of activities, while including more accelerometers on different places, enhances the recognition speed and accuracy. The experiments in this paper show that combining the data from two sensors already gives an enhanced level of context-recognition. Improving the sensors, adding new ones, and adding redundant sensors on different positions on the body, is a logical step we are currently focusing on. We also hope that further research will lead to an improved adaptive system, dealing better with obstacles like the curse of dimensionality and the stability-plasticity dilemma, but remaining on-line and still requiring little user-attention.

We would like to take the experiments further and do some testing over longer periods. Instead of minutes, we would like to see what happens if the hardware is worn for whole days, weeks, or even months. A smaller and less-consuming processing unit would be needed that does the context mapping, since the laptop is rather bulky to wear and its batteries don't last very long. This would be necessary to implement applications such as an automated diary that stores and links the activities a user does during the day. It would also allow a more reliable evaluation on the performance and behavior of the Markov Chain as a probabilistic model. We intend to use a PDA instead of a laptop computer, since the software doesn't require a lot of storage or excessive processing power.

5. Conclusions

Context Awareness without an on-line adaptive behavior is very limited. The user should decide which contexts are valuable instead of the device's designer, and the application should be autonomous enough so that it does not become an inconvenience for the user. This paper demonstrates with a few simple sensors that it is possible to obtain a flexible application, requiring minimal user-interaction.

Real-time processing, limited user-feedback and consistent adaptation are harsh constraints for not just the hardware, but especially for the learning system. Several issues remain, though: the *curse of dimensionality* and the *stability-plasticity dilemma* are well-known problems in the machine learning domain, that are (as their bombastic titles might reveal) not easy. We feel that the combination of artificial intelligence and wearable computing is vital for context awareness and hope that this will eventually lead to a powerful, stable system that can deal with a massive number of sensors.

6. Acknowledgements

This paper was written in the framework of Starlab's Technology for Enabling Awareness (TEA [3], sponsored by European Commissions' Esprit program) and I-WEAR projects. We would therefore like to thank the people from Starlab Research, Nokia, Omega Next Generation and TecO (the TEA partners) and the I-WEAR [7] partners for their support, which was vital for this research. We also wish to

thank Walter Van de Velde, the creator and supervisor of both projects, and the anonymous reviewers for their helpful comments on this paper.

7. References

[1] D. Ashbrook, Context Sensing with the Twiddler Keyboard. In *Proc. of the Third International Symposium on Wearable Computers* (ISWC'99), San Francisco, 1999, p.197-198.

[2] A. Dey and G. Abowd, "Towards a Better Understanding of Context and Context-Awareness". VU Technical Report GIT-GVU-99-22. 1999.

[3] Esprit Project 26900. Technology for Enabling Awareness (TEA). http://tea.starlab.net, 1999.

[4] J. Farringdon, A. Moore, N. Tilbury, J. Church and P. Biemond "Wearable Sensor Badge & Sensor Jacket for Context Awareness". In *Proc. of the Third International Symposium on Wearable Computers* (ISWC'99), San Francisco, pp.107-113.

[5] R.M. French, Catastrophic Forgetting in Connectionist Networks. *Trends in Cognitive Sciences, 3*(4), 1999, pp. 128-135.

[6] A. R. Golding and N. Lesh, "Indoor navigation using a diverse set of cheap, wearable sensors". In *Proceedings of the third International Symposium on Wearable Computers*, 1999, pp. 29-36. 1999.

[7] T. Kohonen, *Self-Organizing Maps*, Second Edition, Springer-Verlag Heidelberg, 1997.

[8] I-WEAR, see http://www.iwear.com.

[9] T. Mitchell, *Machine Learning*. McGraw-Hill. 1997.

[10] J. Paradiso and E. Hu, "Expressive Footwear for Computer-Augmented Dance Performance". In *Proc. of the First International Symposium on Wearable Computers* (ISWC'97), Cambridge Massachusetts, 1997, pp.165-166.

[11] J.K. Perng, B. Fisher, S. Hollar and K. S. J. Pister, "Acceleration Sensing Glove (ASG)". In *Proc. of the Third International Symposium on Wearable Computers* (ISWC'99), San Francisco, 1999, pp.178-180.

[12] A. Schmidt, K.A. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven & W. Van de Velde, "Advanced Interaction in Context", in H. Gellersen (Ed.) *Handheld and Ubiquitous Computing, Lecture Notes in Computer Science* No. 1707, Springer-Verlag Heidelberg, 1999, pp. 89-101.

[13] A. Schmidt, M. Beigl, H.W. Gellersen. "There is more to context than location". Interactive Applications of Mobile Computing (IMC98), Rostock, Germany, Nov. 1998. Reprinted in: Computers & Graphics Journal, Elsevier, Volume 23, No.6, December 1999, pp 893-902.

[14] K. Van Laerhoven, "Teaching context to applications". In *Proc. of the Workshop on Situated Interaction*, CHI 2000, The Hague, 2000, pp. 4-6.

[15] C. von der Malsburg, "Self-Organization of Orientation Sensitive Cells in the Striate Cortex", 1973, in G.A. Carpenter and S. Grossberg (Eds.) *Pattern Recognition by Self-Organizing Neural Networks*, MIT Press, 1991, pp. 179-205.

[16] For some examples see the product datasheets of the Sony N50 TFT Displays, the ECO sensors on JVC MultiSystem TVs, the Planar AMLCD displays or the Compaq P110 monitors.