Context Awareness in Systems with Limited Resources

Ozan Cakmakci OZAN.CAKMAKCI@IMAG.FR
Joelle Coutaz JOELLE.COUTAZ@IMAG.FR
CLIPS (Human Computer Interaction Group), University of Joseph Fourier, B.P. 53, Grenoble, 38041, France
Kristof Van Laerhoven KRISTOF@COMP.LANCS.AC.UK
Hans-Werner Gellersen HWG@COMP.LANCS.AC.UK
Ubiquitous Computing (Ubicomp) Group, Lancaster University, LA1 Lancaster 4YR, United Kingdom

Abstract

Mobile embedded systems often have strong limitations regarding available resources. In this paper we propose a statistical approach which could scale down to microcontrollers with scarce resources, to model simple contexts based on raw sensor data. As a case study, two experiments are provided where statistical modeling techniques were applied to learn and recognize different contexts, based on accelerometer data. We furthermore point out applications that utilize contextual information for power savings in mobile embedded systems.

Keywords: context awareness, statistical approach, smart sensors

1. Introduction

This paper aims at high-lighting the benefits of using a well-founded and traditional method from statistics to model contexts, intending to bring the process of context awareness closer to the sensor-level. We consider this approach to be especially beneficial in applications where power-, processing- and memory resources are meager, such as in the domains of ubiquitous, mobile or wearable computing. We start the introduction with giving an overview of context awareness research, followed by similar research in robotics. Finally, a short overview on low-power design clears the way towards this paper's main case study.

1.1 Context Awareness

With the hope of assisting a user continuously throughout the day, for example in aiding of human memory (episodic, semantic or procedural), two major approaches have been proposed. One is to spread the computation and communication resources into the environment [11] and the other is to carry all the computational and communication resources on the body [15]. One could see clothing and the devices we carry with us as a part of the environment, although the resulting nuances of this view are actively debated (i.e. infrastructure reliance and privacy issues). The idea of spreading computation into our environment in order to get it out of the way of the user is often referred to under names such as ubiquitous computing (ubicomp), pervasive computing or disappearing computing.

In the Ubicomp approach, it is often necessary to build ordinary artifacts that contain computation and communication facilities in them (see [10] as an example). Designing ordinary artifacts with electronics embedded in them while avoiding modification of the functional meaning of the object is a big challenge (due to space and energy constraints). In parallel, advances in microelectronics and low-power techniques (such as frequency and voltage scaling [18] as well as more innovative techniques like adiabetic charge recovery [17]) have led to the creation of computers that fit into the wristwatch form factor.

Context awareness can *loosely* be defined as applications or devices having a basic understanding of their user's location or activity over time. An overview on understanding and using context is provided by Dey in [21]. Early work in context awareness (in the mobile computing community) focused on location aware systems. Examples of location aware systems include the Active Badges work at Olivetti [16] and Pepys automatic diary generator by Lamming [19]. Most of these systems relied on infrared transceivers to detect the physical location.

More recent research is beginning to address more subtle types of context such as trying to infer psychological state from physiological measurements [8] and using various sensors to infer activity of a user. Physical sensors play a key role in collecting the data that leads to understanding the user behavior. Typically, machine learning techniques are utilized to extract context information from sensor data. For a neural net based approach that uses low-level sensors, see [1], for a hidden markov model based approach that uses a combination of low-level sensors with cameras and microphones, see [2].

1.2 Context in AI and Robotics

The robotics community has been working on similar problems. Early work in context in artificial intelligence goes back to Kant's schemas, Minsky using frames [13] for scene analysis, work by Hanson and Riseman [12] and use of context for object recognition by Strat and Fischler [14]. At the implementation level, when thinking about extracting a user's context from sensor data, we may resort to a proposal made within the robotics community (see [6]) where the authors suggest custom designed sensors that preprocess the data and central processors that operate on high-level features rather than raw data.

1.3 Low Power Design

The electronic systems we are building today to realize the vision of ubiquitous devices rely heavily on CMOS (Complementary Metal Oxide Semiconductor) technology. Therefore, the characteristics of these systems starting at the physical level directly influence higher levels of design, for example the perceived user models of a device, especially in mobile systems.

We will briefly discuss dynamic power consumption in CMOS circuits and point out how contextual information could help in reducing power. Dynamic power consumption in CMOS circuits is characterized with:

$$E = CL*VDD2*f^2 \quad \textbf{(1)}$$

with CL the load capacitance, VDD the operating voltage, and f the frequency of operation. In order to lower dynamic power, one could reduce the capacitance, operating voltage, or the frequency. Heuristics such as powering down parts of the chip that are not in use are also utilized in order to save power. In fact, one could view this paper as providing these higher level heuristics linking system activity and actions to the user activity.

The remainder of this paper contains a theoretical introduction explaining the Bayesian background of the approach, after which a toy problem from the wearable computers community should illustrate its use and simplicity. Preliminary experiments in the form of a case study serve as a starting point in the general class of consumer electronics, especially on applications where context information may result in power savings and potentially other benefits that could enhance user experience and interaction with these devices.

2. Statistical Modeling of Context

Fundamental to our approach is the modeling of contexts with statistics from training data. We are characterizing each context with samples of sensor data taken from each context to estimate probability densities corresponding to contexts. Once we have an estimate for the density characterizing a context, we can proceed to compute the probability of being in a certain context given sensor data, i.e., p(context | sensor data). We know from Bayes' rule that

$$p(context \mid sensordata) = \frac{p(sensordata \mid context) * p(context)}{p(sensordata)} \quad \textbf{(2)}$$

We will assume that each context can be characterized with a normal density with mean **μ** and covariance matrix **Σ**. It follows from this assumption that the maximum likelihood estimates for parameters mean **μ** and covariance matrix **Σ** is as follows [3]:

$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k \quad \textbf{(3)}$$

$$\hat{\Sigma} = \frac{1}{n}\sum_{k=1}^{n}\left(x_k - \hat{\mu}\right)\left(x_k - \hat{\mu}\right)^T \quad \textbf{(4)}$$

Once we have the $\hat{\mu}$ and $\hat{\sum}$ estimates representing the distributions of each sample class, we can compute the Mahalanobis distance, in order to classify sensor data in relation to the modeled data. Given sensor data, we will choose the context class where the sensor data has a maximal probability of belonging to that class, i.e.

$$\max_k\left(p(context_k \mid sensordata)\right) = \max_k\left(p(sensordata \mid context_k) * p(context_k)\right) \quad \textbf{(5)}$$

At this point, we can substitute our previously assumed normal density, i.e. p(sensordata | context) ~ N(μ,Σ), and we will take the logarithm of both sides and classify by taking the maximum for the log-like likelihood of each context class. This will make the mathematical manipulations easier (since a Gaussian is of form e^x) while not changing the meaning of the equation (in terms of its maxima) due to the log function being monotonic.

$$\max_k\left(p(context_k \mid sensordata)\right) = \max_k\left(\ln\left(\frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_k|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1}(x-\mu_k)} \left(p(context)\right)\right)\right) \quad \textbf{(6)}$$

Which leads to:

$$\max_k\left(-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1}(x-\mu_k) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma_k| + \ln p(context)\right) \quad \textbf{(7)}$$

We can ignore the additive constants in equation 6 such as the $(d/2)\cdot\ln(2\pi)$, ln p(context) in the case where all contexts are equally likely, and the determinant of the covariance matrix $(1/2\cdot\ln|\Sigma|)$. We will be left with the maximum of the negative Mahalanobis distance, or the minimum Mahalanobis distance:

$$\max_k \left(p\left(context_k \mid sensordata\right)\right) = \min_k \left((x-\mu_k)^T \cdot \Sigma^{-1} \cdot (x-\mu)\right) \quad \textbf{(7)}$$

3. Toy problem– detecting physical contexts such as sitting, standing and walking

In order to clarify the technique, we first apply statistical modeling to a toy problem where we try to distinguish the physical contexts sitting, standing and walking. Figure 1 shows a plot of the acceleration data of 4500 acceleration data points over time, where the subject mainly changes between sitting, standing up, and walking (see [1] for more details on the data set).

For training, we have used the first 1000 data points of the initial data set where the user is sitting, standing and walking (in that order). See Figure 2. We have computed the mean μ and covariance Σ for sitting, standing and walking regions, giving us the following values:

μsitting = [58.9533 71.2633]

Σ sitting = [0.3391 0.0229 ; 0.0229 0.3391]

μstanding = [70.3920 57.7807]

Σ standing = [0.3391 0.0229 ; 0.0229 0.7651]

μwalking = [67.3425 56.7845]

Σ walking = [131.0265 78.2076 ; 78.2076 64.7255]

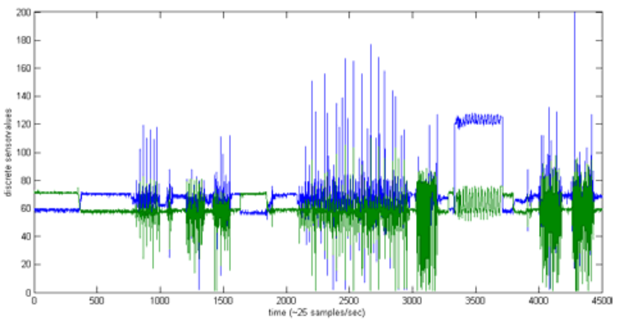

Figure 1. Time series plot of the entire experiment.

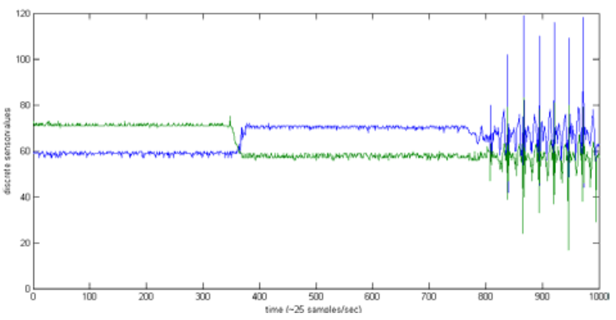

Figure 2. First 1000 samples, used as training data.

We have computed the data shown in Figure 1 as a test set, using the mean μ and covariances Σ for sitting, standing and walking regions.

The data shown in Figure 1 was used to recognize the contexts training set to calculate the Mahalanobis metric. A table of recognition results is presented below in Table 1. For the sake of space, we have averaged the recognition result of each activity. For example, if sitting occurred five times during the experiment, we have checked with what probability the sitting occurs in each of these regions and averaged the results.

Table 1. Average probabilities per context

Activity	Recognition rate
Sitting (occurs 3 times during the experiment)	95.66%
Standing (occurs 2 times during the experiment)	80%
Walking (occurs 9 times during the experiment)	93.11%

4. Case study – detecting when users glance at their watch

We provide in this paper a case study where a user's context, related to a power saving application in this instance, would be whether a user is looking at their watch or not. Context information can be used to automatically shut off subsystems (such as the display) and could facilitate further power savings within the sleep mode. If we can appropriately detect when the user is glancing at their watch, we can automatically activate the sub-systems (such as the display) and avoid user annoyance when they look at their watch and not see it there.

As noted by the designers of the IBM wristwatch computer, several challenges exist in this form factor, power being a major one. To take the IBM wristwatch as a concrete example, it consists of the following sub-

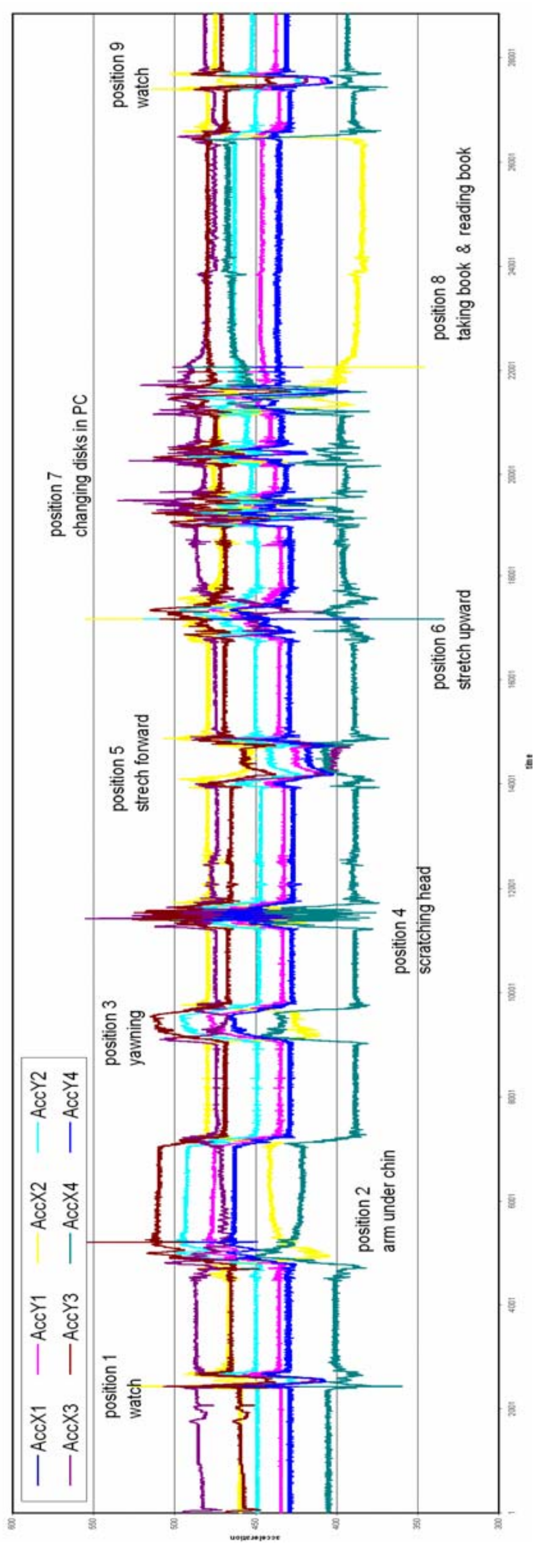

Figure 3. Evaluation set, used in a blind test experiment.

systems: an ARM7 processor, 8Mb flash memory, 8Mb of DRAM, serial, IRDA, and expansion interfaces. Designers have tested a reflective LCD as well as an organic LED. We should note that IBM is designing the newer version of their watch to include an accelerometer sensor[. Their paper discusses several optimizations on how to save power in active and sleep modes and conclude in the future work section with "*when the system is in sleep mode, most of the power is consumed by the display and the DRAM refresh*".

Looking at a watch gesture we will consider here consists of lifting of the arm, holding the arm steady in order to read the contents of the display, and then releasing of the arm. Figure 3 shows real acceleration data for this process while the user is standing.

It is important to mention that the current user model for a watch includes replacing the battery of the watch in a time scale of years. Information appliances such as PDAs or laptops require charging of batteries almost on a daily basis. Therefore, it is challenging to provide functionality similar to a PDA or a laptop in the watch form factor while retaining the user models.

4.1 Modeling the watch data

We have, in terms of modeling the data, collected accelerometer data while a user is glancing at their watch during various office settings such as when the user is sitting or standing. The mean (**μ**) and the covariance matrix (**Σ**) were collected over the whole dataset where a user is glancing at their watch. Within the statistical pattern recognition literature, this approach is known as the single hypothesis case, where we have a single well defined class while other classes are not [4]. We should note that for low dimensional data (such as a single accelerometer with 2 or 3 dimensions), a single hypothesis scheme is expected to work well, however as dimensions increase, the error of this technique increases as well.

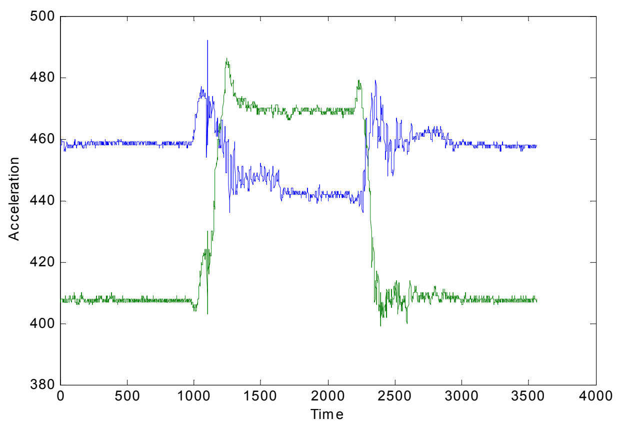

Figure 4. A typical example of training data for looking the watch gesture.

Our training dataset included five datasets of looking at the watch gestures. Figure 4 depicts a plot of one of these datasets. Three different regions in this dataset stand out: The first region appears as a stable state while the arm assumes posture as seen in Figure 6. The second region is while the user is glancing at his watch and the third region is back to the standing posture as seen in Figure 7.

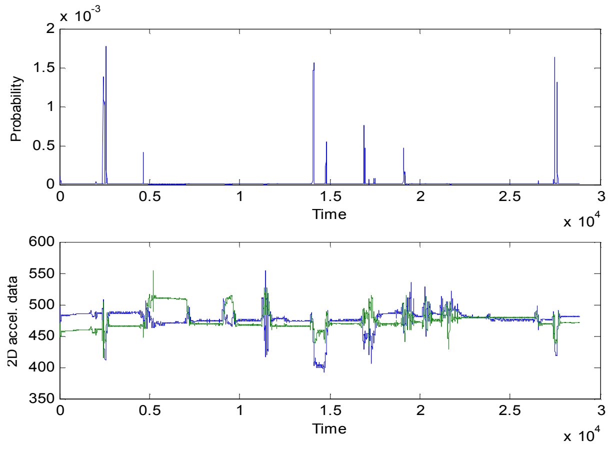

Figure 5. The results of the experiment: the first plot (top) shows the probability of each point in the test set (bottom) belonging to the looking at the watch class. Only one dual-axis accelerometer was chosen from the evaluation set in

Figure **3**.

For modeling the data, we have extracted the middle regions from each of the five datasets and computed the statistics over the whole training samples. We can use this learned distribution to evaluate the probability of looking at the watch on a test set. Our test set is shown in Figure 3, the probability of each point in the test set belonging to the looking at the watch gesture is shown in Figure 5, both taken at roughly 220 samples per second.

4.2 Assessment of results

Based on the experiment results in Figure 5, one can see that the system can recognize properly when the user is looking at their watch on the test set. However it gets confused, for example, when the user is scratching his head. We believe that we can make the recognition more robust by computing statistics over features as opposed to raw sensor values. Direct disadvantages of calculating features for the kind of applications that we envision would be the delay of recognition that such an approach could bring along, and the harsher memory requirements.

We would furthermore like to note that this approach proved to be reasonably robust when it was evaluated in the case the user was standing up, or even lying down. Prerequisite is that these are also trained beforehand and incorporated by means of an additional mean and covariance matrix. Real-life feasibility studies with multiple users are likely to be required, however, to evaluate the attractiveness of this particular application.

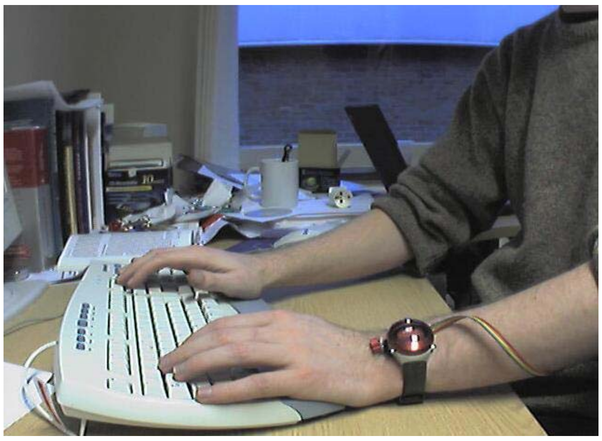

Figure 6. Typical office setting (not glancing at the watch). Connected to the watch is a small board containing the dual-axis ADXL202 accelerometer.

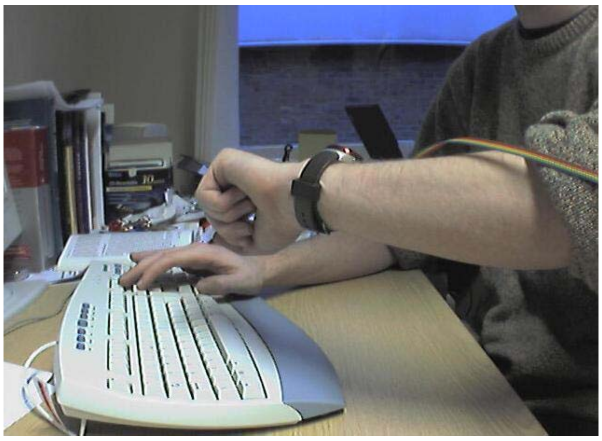

Figure 7. The wearer glancing at his watch.

As is stated in the future work section of [7], "*by definition we did not want the display to be turned off since it is annoying to glance at your watch and not see the time.*". This assumes that there is no robust higher level knowledge whether the wearer is looking at the watch. The results in this paper contest the unfeasibility of this by using a combination of a microcontroller and accelerometer that is comfortable in power consumption.

5. Implementation

We deem that parametric statistical modeling techniques present some advantages that make them suitable for implementation in systems with scarce resources. The statistical techniques presented here, have been implemented on a PIC microcontroller with limited memory and computational capabilities.

We have evaluated the Mahalanobis distance using the mean vector and covariance matrix from the training set, and utilized the Analog Devices ADXL202 sensor in our experiments. This sensor can measure both dynamic acceleration (e.g., vibration) and static acceleration (e.g., gravity). The sensors ability to measure gravity gives us the opportunity to discriminate in contexts where acceleration may be zero (such as sitting and standing). The sensors' ability to measure dynamic acceleration gives us the ability to have an idea about whether the user is walking or running, but also hints when the user is moving the watch to a glancing position.

5.1 Time and Space Complexity

In the general case, the number of parameters for specifying a multivariate Gaussian distribution is described as follows:

$$N_{\text{parameters for a context}} = d + \frac{d * (d+1)}{2},$$

where d stands for the number of dimensions. This suggests that for 2 dimensional acceleration (acceleration in X and Y directions) data we would expect to have 5 parameters. These parameter will include, in the two dimensional case, a mean vector, i.e., $\mu = \{\mu_1, \mu_2\}$, and a covariance matrix, i.e.

$$\Sigma = \begin{bmatrix} \sigma_1^2 \sigma_{12} \\ \sigma_{21} \sigma_2^2 \end{bmatrix}$$

Since we need these parameters for modeling each context, the total number of parameters will equal ($N_{\text{parameters}}$* number of contexts). In our case study, we have trained for just one contexts (glancing at a watch), therefore we just need 5 parameters. The complexity hence increases linearly as the number of modeled contexts goes up.

In section 3, we have shown that once the contexts are represented we would need the Mahalanobis distance in order to classify given data to a context. In order to compute the Mahalonobis distance on board, we require mean and covariance matrix of training data. One could calculate the mean and covariance per context on the microprocessor as a calibration method, depending linearly on the length of the interval. If the sensors are adequately calibrated, however, these parameters could be stored as constants.

5.2 Pseudo-code for micro implementation and cycle cost estimates for different architectures

Assuming that the training was done beforehand, each context's model has to be stored as a mean vector, plus a covariance matrix. In our experiment, we have only one significant context, glancing at one's watch, and just two-dimensional input vectors to deal with, resulting in a comfortable memory situation, even for today's smaller microcontrollers.

The source code itself is very trivial; without the necessity of determining the Gaussian for the given input, and instead calculating the Mahalanobis distance, we simply need to concentrate on subtraction, addition and multiplication:

```
//  constant  values  for  the  means,
// acquired from training data:
const float mu1=441.9091;
const float mu2=456.1567;

// the elements of the inverse of the
//  covariance  matrix,  also  acquired
// from training data:
const float sig11=0.0079;
const float sig12=0.0045;
const float sig22=0.0184;

// elements of Mahalanobis formula:
float mx1, mx2;
mx1 = x1 – mu1;
mx2 = x2 – mu2;
// calculate 2d Mahalanobis:
mah_dist = mx1*(mx1*sig11+mx2*sig12)+
           mx2*(mx1*sig12+mx2*sig22);
```

In case the calculation of the mean vectors and covariance matrices is done on the microcontroller, additional memory might be required as a storage medium to increase reliability, but on-board processing produces no real problems[1] for current microcontrollers.

The basic (non-assembly-optimized) version of our microcontroller software takes 892 bytes of memory, mostly because it requires supporting routines for floating points (this is just 11% of our microcontroller's ROM). In worst case (as calculated by the CSC C compiler), 41 bytes (or 23%) of the microcontroller's RAM memory will be used.

[1] Source code for the PIC16F877 in CCS C is available at http://www.comp.lancs.ac.uk/~kristof/notes/ as this is too lengthy to publish in this paper.

5.3 Power measurements

One of our claims has been that context information can contribute to power savings. We will give in this section a detailed contemplation of how our approach would benefit and have an impact on mobile and wearable applications, where power conservation is imperative. We will do this again using the illustration of the watch.

Our current prototype is based on an off-the-shelf ADXL evaluation board (i.e. MPLAB-ICD DEMO BOARD revision 1). It draws .25mA@9VDC. In other words, due to Power (Watts) = I (Ampere) * V (Voltage): 125mW. We believe we would need to go down orders of magnitude from this upper limit to provide useful services. Several improvements to this figure are possible. The current demo board includes a voltage regulator, a PIC16F877 microcontroller, eight LEDs and a maxim voltage level conversion chip. The microcontroller in the demo board is clocking at 20Mhz.

If we take the current microcontroller and design a custom board (running at 3V and clocking at 32Khz) only with a PIC 16F877 an ADXL202 + passives, we could readily improve this figure. The power consumption for the PIC would go down to 60µW and we would need to add the ADXL to this figure. Note that this particular microcontroller will draw 20µA@3V, operating at 32Khz. Each instruction will take 125µ sec at this operating frequency.

This particular PIC microcontroller has a SLEEP mode where it consumes less than 1uA at the given operating voltage. We could assume 3V for the custom designed board. When sampling at 20Hz., the resulting interval between two samples would be 50ms. The PIC 16LF877 executes each instruction in a single cycle. Each cycle consists of four clock oscillation periods. In the case of 32Khz, instruction execution time is 125µs (1/f*4). We have compiled our code with the CCS C compiler and seen that it requires approximately 800 instructions of computation. Let's define "C" as mean computation time and we will have C = number of instructions * 125µs.

Evaluating the Mahalonobis distance on a new set of data takes approximately 100ms. In order to meet our 20Hz sampling specification, we need to either increase the frequency or optimize our code through handwritten assembly. Here we will present expected result by almost tripling the frequency to 100Khz. The datasheets suggest a linearly relation between current draw and frequency for a given voltage, drawing 80µA@3V at the new frequency. Therefore, the power consumption will be around 240µW. The expected consumption should be less than this due to utilizing the sleep modes while waiting for the next sample to arrive, however, an exact calculation requires knowledge of time frame of the power measurement in (W/t) and we could not find this in the datasheets.

6. Alternative Recognition Techniques

We have considered using the maximum likehood estimates for the densities representing contexts. Several alternative recognition techniques could be considered. For example, in some previous experiments, we have experimented with the self organizing map (SOM). Though it is considered as a fast-acting algorithm with few resource-constraints, implementing the SOM on a microcontroller will limit its performance dramatically. We achieved maximally a one-dimensional map of 25 units, which has proven to limit the capabilities of the SOM especially in the number of target contexts.

One could also pose this problem as a recognition of the sequences between the states, in which case models such as hidden Markov models (HMM) could be useful. An HMM is defined compactly with the parameters $\lambda = (\Pi, A, B)$, where Π represents the initial distribution, A represents the transition matrix and B represents the output probabilities at each state. For comparison purposes, we can assume that the training of these model parameters will be done beforehand and only the Viterbi decoder will be implemented on the microcontroller. Computational requirements for Viterbi decoding depends on the length of the sequence and a simple decoder will have a complexity of $O(c^2T)$, where c is the number of states and T is the length of the sequence[3]. We have not modeled this data using HMMs, however, our feeling (by looking at problems HMMs have been applied to) is that between 1-3 states should be sufficient to model this data with HMMs. In the lower end of the state count, the number of operations could be relatively similar; however, the number of operations will increase as more states are needed.

7. Conclusions

We believe that it is important to incorporate data processing closer to the sensors in the signal flow path between processor(s) and sensor(s) in order to save power, cost and meet real-time operation requirements. Using modeling techniques from standard statistics, simple contexts can be distinguished at such a level. Although this could be helpful to any application, it is especially valid in the case of mobile or wearable systems.

Given the clear trend in combining communication and computation capabilities with sensors, see [5] and [9], algorithms that are sensitive to power (both computational and electrical) will play a role when we distribute several smart sensors around our bodies or devices within our environment. In this case, distributing computation will also reduce the load on a central computer (if one exists) and makes the system robust to failures. We have presented some initial ideas on how contexts provide added value to a system and discussed techniques to achieve that efficiently.

Acknowledgements

We would like to thank Francois Berard, Jim Crowley, Daniel Hahn, and Martin Strohbach, for their support, and especially Albrecht Schmidt for assistance in the hardware development. Kristof Van Laerhoven is funded by the Equator IRC, supported by EPSRC. Ozan Cakmakci is funded by GLOSS, supported by the EC's Disappearing Computer initiative.

References

[1] Van Laerhoven, K. and Cakmakci, O., "What Shall We Teach Our Pants?," *Proc. 4th Int'l Symp. Wearable Computers (ISWC 00)*, IEEE CS Press, Los Alamitos, Calif., 2000, 77-83.

[2] Clarkson, B., K. Mase and A. Pentland. Recognizing User Context via Wearable Sensors. Proc. 4th Int'l Symo. Wearable Computers (ISWC 00), IEEE CS Press, Los Alamitos, Calif., 2000.

[3] Duda, Hart and Stork. *Pattern Classification*. Wiley. 2001.

[4] Fukunaga, K., Introduction to Statistical Pattern Recognition. Morgan Kaufmann. 1990.

[5] Lee, K. "IEEE 1451: A standard in support of smart transducer networking". IEEE Instrumentation and Measurement Technology conference, Baltimore, MD, USA, May 2000.

[6] Etienne-cummings, R. "Intelligent robot vision sensors in VLSI". Autonomous Robots 7, 225-237, 1999.

[7] Kamijoh, et.al. "Energy trade-offs in the IBM Wristwatch computer". *Proc. 4th Int'l Symp. Wearable Computers (ISWC 01)*, IEEE CS Press, Los Alamitos, Calif., 2000, 133-140.

[8] R. Picard. Affective Computing. MIT Press, 1997.

[9] Holmquist, L.E., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M. and Gellersen, H.W. Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts, Proc. of UBICOMP 2001, Atlanta, GA, USA, Sept. 2001.

[10] Gellersen, H.W., Beigl, M, Krull, H. The MediaCup: Awareness Technology embedded in an Everyday Object. In H. Gellersen(Ed.) *Handheld and Ubiquitous Computing*, Lecture Notes in Computer Science No. 1707, Springer-Verlag Heidelberg: 1999.

[11] Weiser, M. "The Computer for the 21st Century." *Scientific American*, Vol. 265, No. 3, September 1991.

[12] Belkna, R., Riseman, E., and Hanson, A. "The information Fusion Problem and Rule-Based Hypotheses Applied to Complex Aggregations of Image Events", *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 227-234, 1986.

[13] Minksy, M. The Psychology of Computer Vision, P.Winston (Ed.), McGraw-Hill, 1975.

[14] Strat, T.M., and Fischler, M.A., *Context-Based Vision: Recognizing Objects Using Information from Both 2-D and 3-D Imagery*, *PAMI(13)*, No. 10, October 1991, pp. 1050-1065.

[15] Starner, T.,S. Mann, S.,B. Rhodes, B.,J. Levine, J., Healey, J.,D. Kirsh, D.,R. Picard, R.,A. Pentland, A., "Augmented Reality Through Wearable Computing," Presence: Teleoperator and Virtual Environments, Vol. 6, No. 4, pp. 386-398, August 1997.

[16] Want, R.,A. Hopper, A.,V. Falcao, V., and J. Gibbons, J. . "The active badge location system..". ACM Transactions on Information Systems, 10(1):91-102, January 1992.

[17] Athas, W.C., "Energy-recovery CMOS". In Rabaey, Pedram (Eds), *Low Power Design Methodologies*, Kluwer, 1996.

[18] Pouwelse, J., Langendoen, K., Sips, H., Voltage scaling on a low-power microprocessor, J. Pouwelse, K. Langendoen, H. Sips, Mobile Computing Conference (MOBICOM), Jul 2001

[19] Lamming M. G. and W. M. Newman, "*Activity-based Information Retrieval: Technology in Support of Personal Memory*", in Proceedings of 12th World Computer Conference, 1992.

[20] IBM wristwatch computer project webpage. March 22,2002. http://www.research.ibm.com/WearableComputing/collaboration/ibmcitizen.html

[21] Dey, A.K.Understanding and Using Context Personal and Ubiquitous Computing Journal, Vol. 5 (1), 2001, pp. 4-7.