Self-Organization in Ad Hoc Sensor Networks: An Empirical Study

Elaine Catterall, Kristof Van Laerhoven and Martin Strohbach
Computing Department
Lancaster University
Lancaster LA1 4YR, United Kingdom
{elaine,kristof,strohbach}@comp.lancs.ac.uk

Abstract

Research in classifying and recognizing complex concepts has been directing its focus increasingly on distributed sensing using a large amount of sensors. The colossal amount of sensor data often obstructs traditional algorithms in centralized approaches, where all sensor data is directed to one central location to be processed. Spreading the processing of sensor data over the network seems to be a promising option, but distributed algorithms are harder to inspect and evaluate. Using self-sufficient sensor boards with short-range wireless communication capabilities, we are exploring approaches to achieve an emerging distributed perception of the sensed environment in real-time through clustering. Experiments in both simulation and real-world platforms indicate that this is a valid methodology, being especially promising for computation on many units with limited resources.

Introduction

It is obvious that distributed sensing has been inspired to a great extent by biological systems, where highly redundant sensors appear in the form of *duplicate sensing* (e.g. having two eyes), *fusory sensing* (e.g. seeing and touching the same object), and *distributed sensing* (e.g. networks in the skin) (Brooks 1988). It is our aim to investigate what the impact of distributed algorithms, self-organization, and sheer number (scalability) of sensor modules will have on sensing and perception of the environment.

Moving sensing tasks to real-world applications often proves to be impractical, as the concepts that have to be learned and distinguished are too complex to be captured by just a few sensors. Instead of improving the quality of the used sensor(s), the quantity is increased in distributed sensing. Benefits of this approach have been mentioned early on in sensor fusion literature (Ayache 1990; Brooks & Iyengar 1998):

1. redundancy in sensors leads to a more robust system since faulty sensors have little effect on the output,

2. distributed sensors have a higher chance to capture relevant aspects because of their spatial spreading, and

3. the cost of producing many sensor modules that perform recognition concurrently is considered to be smaller, since sensors can be smaller and are not required to be as precise.

We will concentrate in this paper on clustering data originating from a numerically fixed set of sensors, concurrently operating in the same environment. A prime requirement is that the clustering should be done in a decentralized way, since it is being implemented on a hardware platform based on microcontrollers with limited memory. The output of this distributed sensor network is the distributed storage of typical representations of various states, or contexts, of the environment.

Research into new interaction techniques in ubiquitous computing is gradually moving towards 'smarter' objects that are able to monitor their environments with hardware sensors. This research is usually referred to as 'context awareness' (Abowd, Dey, & Brotherton 1997). The many sensors approach in context awareness has attracted attention from various research domains (Kahn, Katz, & Pister 1999; Lim 2001).

The Hardware Platform

The platform for the experiments in this paper is a collection of 'Smart-Its'[1]. In this section we describe some of the characteristics of the Smart-Its in order to provide insight into the experimental setup, and illustrate some of the limitations. One Smart-It unit embodies a *sensor module*, and a *communication module*, which are interconnected.

The core of sensor module is a PIC 16F877 microcontroller clocked at 20 MHz, which offers 384 bytes of data memory and 8K×14 bits of memory. See Figure 1 for the arrangement of the sensors. A library comes with the module that provides easy access to the sensor values. A serial line connector is available for connecting the sensor module to a PC. Of the two I^{2}C connectors, one is used to interface to the communication module.

[1]Smart-Its Project Home Page. http://smart-its.teco.edu/

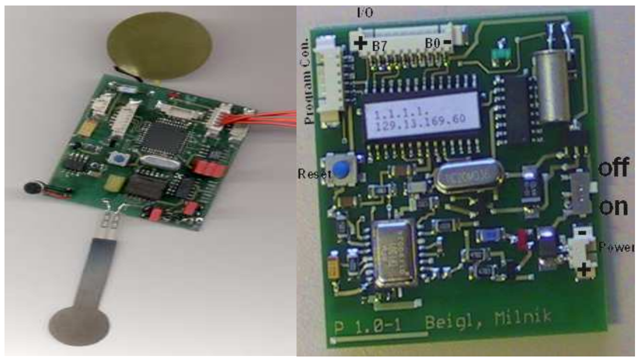

Figure 1: The two modules that combined make up the Smart-It unit: Left, the sensor module with light sensor, microphone, 2 accelerometers, thermometer, pressure sensor (bottom) and buzzer (top). Right, the wireless communication module. Both boards measure 43 × 50 mm.

The communication module is based on the PICF876. An RF stack provides wireless communication, at a maximum rate of 125 kbit/s. The current implementation of the RF stack only supports broadcast. Two of the I/O pins are used for communication with the sensor module over I^2C. The I^2C interface offers read and write access to field strength information, as well as access to the sender's IP and the application data packet.

The Kohonen Self-Organizing Map

Self-organization of neuronal functions seems to exist on very abstract levels in the brain. When a laboratory rat has learned its location in a labyrinth, certain brain cells on the hippocampal cortex respond only when it is in a particular location. The Kohonen Self-Organizing Map (SOM) (Kohonen 1997) has a similar principle: units (referred to as neurons) are recruited topologically for tasks depending on the sensory input. It is commonly classified as a neural network, and more specifically a winner-takes-all competitive algorithm, since the units *compete* with each other for specific tasks.

Each unit i has its own prototype vector w_i (also referred to as codebook vector or weight vector), being a local storage for one particular kind of input vector that has been introduced to the system. Initially these prototype vectors with a dimension n equal to the input space, start out as vectors with random small components and, as new input enters the SOM, are improved following this update rule:

$$w_i = w_i + \alpha \cdot \eta(\text{winner}) \cdot (x_i - w_i), \forall i \in \{1, \ldots, n\}$$

where α is called the learning rate and is situated between 0 and 1, and $\eta(\text{winner})$ is the neighbourhood function ranging from 0 to 1 as well, depending on the distance between the current SOM unit and the winner.

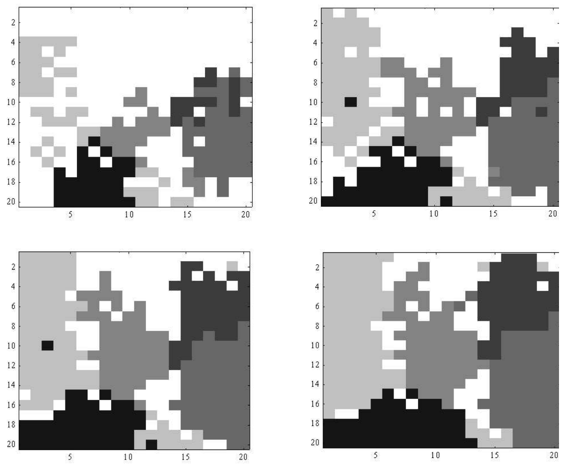

Figure 2: A 2D Self-Organizing Map showing different regions of winning units for different states of the environment.

The winner is the unit that has a prototype vector that is closest to the current input vector using the Euclidean distance:

$$\text{winner} = \arg\min_j \sqrt{\sum_{j=1}^{n} (x_j - w_j)^2}$$

The neighbourhood function is traditionally implemented as a Gaussian (bell-shaped) function:

$$\eta(\text{winner}) = \frac{1}{\sqrt{2\pi}nb} e^{-0.5\cdot(\text{winner}-\text{current})^2/nb^2}$$

with nb a parameter indicating the width of the function, and thus the radius in which the neighbours of the winning unit are allowed to update their prototype vectors significantly. The map of units is usually taken as a two-dimensional grid, although many other organisations have been applied (such as a map of hexagons).

After a sufficient amount of input data has been presented to the SOM, self-organization will result in a topographic map, where similar data is mapped onto units in a particular region of the map, and neighbouring units will be activated (i.e. become winners) for similar input data. Figure 2 shows how different units become recruited for different states of the environment by colouring the units according to the state in which they were declared as winners.

Implementation

The Kohonen SOM mainly has implementations based on a single-processor, centralized method. Therefore, it is necessary that we elaborate on the implementation for the smart-its platform and the simulation[2].

[2]The source code and data files are available for download from http://www.comp.lancs.ac.uk/˜catterae/alife2002

1	Unit ID
2	Packet Timestamp
3	Error (Euclidean distance between prototype and input)

Table 1: Packet description for the Kohonen Self-Organizing Map implementation.

The distributed implementation of the Kohonen Self-Organizing Map (SOM) produces an algorithm that has several variations from the traditional centralized version:

- The units of the SOM are embodied by the Smart-Its: each Smart-It records a prototype vector that resembles one particular kind of input it has experienced. Note that this creates little stress on the resources, since one vector easily fits into the microcontroller's memory.
- The topology of the SOM, instead of being a fixed grid, has a loose topology, defined by the physical distances between the Smart-It units.
- The input for each of the units is different, though similar since it comes from readings from the same state of the environment. In a traditional SOM however, inputs for all units are exactly the same.
- Units can be moved, removed, or added, resulting in a truly ad-hoc network. This resembles research done on growing SOMs and SOMs where 'dead units' (i.e. units that never tend to win) perish (Fritzke 1997).

Communication between the units across the network consists of packets that encapsulate all the necessary information to complete both the find-winner and the update-prototype-vectors phases. After having read the sensor values, each unit compares those values with its internal values, stored in the randomly initialised prototype vectors and calculates the Euclidean distance between both vectors. A packet is then created and broadcast across the network with the elements as they are listed in Table 1. The timestamp is provided to eliminate outdated packages.

After receiving packets from the units in its neighbourhood, a Smart-It can identify the winner by searching for the minimum error. By then calculating how close it is to the winner (in physical distance), the prototype vector can be updated as shown in the update rule above.

The output of the network is the 'activation' of the winning unit, which will be consistent with a self-organized topology, provided enough iterations have been introduced to the network, or if the network does not change too rapidly.

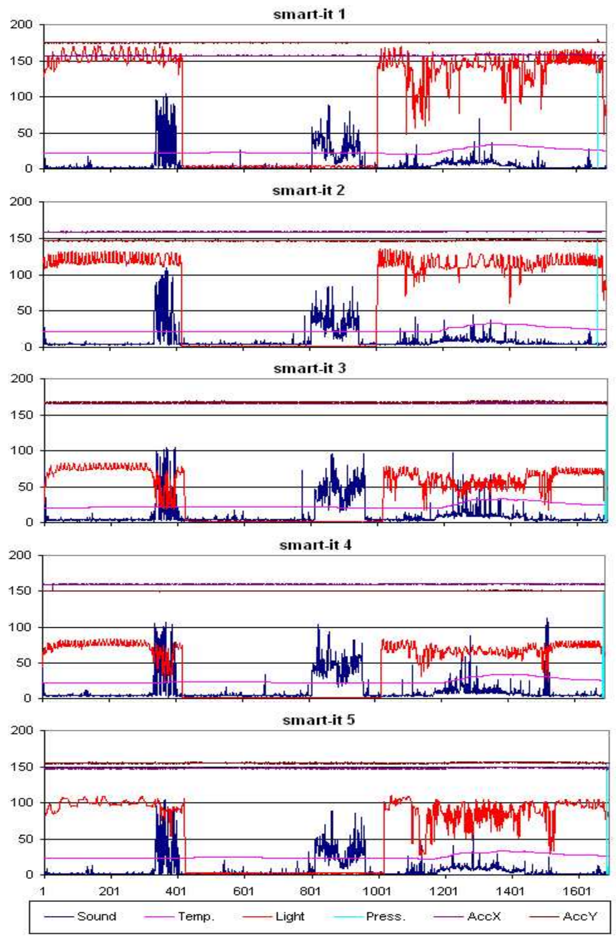

Figure 3: Datasets with sensor values from each of the smart-it units during several states of the environment ('contexts'): Lights on (1–330), talking people nearby while lights remain on (331–400), lights turned off (401–800), talking people nearby while light remain turned off (801–1000), and heating on (1090–1400).

Discussion

Experiments on both simulation and Smart-Its platform are discussed in this section. All results presented here were produced using datasets containing real-world data from the actual sensor modules, and executed on the simulation platform that was designed to be as close as possible to the hardware units. This was done to allow us to evaluate the impact of varying the learning rate and neighbourhood radius parameters, whilst using the same sets of sensor data.

The five datasets (one for each smart-it) are visualized by time series plots in Figures 3. Note that, although the sensor data is very similar (as the units are physically close to each other), it is not exactly the same. The intensity of the light for instance (marked in the legends by 'Light'), is higher for units 1 and 2 since they were positioned directly underneath the light source. The accelerometer data is also a bit different as not all boards were positioned in the same fashion (the readings from the accelerometers reflect position in the X-axis and Y-axis). Also notice how some sensors, such as the temperature sensor, change only slightly and gradually, while others such as the sound level, tend to vary a lot. The pressure sensor on each unit was used to synchronize the data amongst the smart-its, as can be observed around sample 1680 on the X-axis.

The success of our implementation of the Kohonen Map, as on any of its implementations, depends heavily on the choice for the learning rate and neighbourhood radius parameters. We will concentrate on the first, as it is the most important one and the number of smart-it units is not large enough to evaluate the impact of a changing neighbourhood radius.

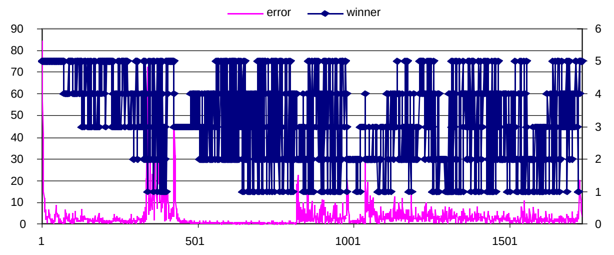

Figure 4: Error (left Y-axis) and winning unit IDs (right axis) over time with a high learning rate. Units ‘forget’ the stored prototype vector, which gets overwritten by the current input.

Figures 4 and 5 show the behaviour of the SOM with a different learning rate. With a high learning rate, units easily overwrite their prototype vector with each new kind of input vector. This problem is known in machine learning as ‘catastrophic forgetting’ and relates to the ‘plasticity-stability dilemma’ (Grossberg 1976). Results with a smaller learning rate (Figure 5) do preserve their prototype vectors, although fluctuating sensor data causes the recruitment of several units for one context (331–400 and 801–1000 for instance, where the sound level varies heavily).

The results from our experiments show that self-organization does take place and that similar sensor data maps onto sensor units topographically. Similar data clusters in a particular region of the environment populated with sensor units.

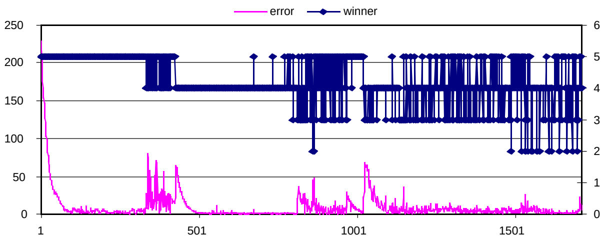

Figure 5: Error (left Y-axis) and winning unit IDs (right axis) over time with a normal learning rate. Unit 5 specializes for the first context, unit 4 specializes for the third context, while the other two contexts are not introduced long enough to be claimed by one unit.

Conclusions

Our aim in applying artificial life principles to the domain of context clustering and eventually context classification and discovery is to provide flexibility and robustness in a constantly changing environment. The sensor boards themselves are relatively simple. By harnessing their collective intelligence arising from their interactions, we aim to produce systems where individual sensor boards and/or sub networks of sensor boards in the collection, learn to specialize in recognizing a particular state of the environment or context.

Our experiments demonstrate that clustering of incoming sensor data through self-organization on many distributed sensor modules with limited processing capabilities is possible.

Acknowledgements

We would like to express our gratitude to our colleagues in the Smart-Its project, sponsored by the Information Systems and Technology framework of the European Commission.

References

Abowd, G.; Dey, A.; and Brotherton, J. 1997. Context awareness in wearable and ubiquitous computing. In *Proceedings of the First International Symposium on Wearable Computing (ISWC)*, 179–180. Boston, M: IEEE Press.

Ayache, N. 1990. *Stereovision and sensor fusion.* MIT Press.

Brooks, R. R., and Iyengar, S. S. 1998. *Multi-Sensor Fusion.* Prentice Hall.

Brooks, M. 1988. Highly redundant sensing in robotics — analogies from biology: Distributed sensing and learning. In *Proceedings of the NATO Advanced Research Workshop on Highly Redundant Sensing in Robotic Systems,.*

Fritzke, B. 1997. Some competitive learning methods. Technical report, Artificial Intelligence Institute, Dresden University of Technology.

Grossberg, S. 1976. Adaptive pattern classification and universal recoding, I: Parallel development and coding of neural feature detectors. *Biological Cybernetics* 23:121–134.

Kahn, J. M.; Katz, R. H.; and Pister, K. S. J. 1999. Mobile networking for smart dust. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 99).*

Kohonen, T. 1997. *Self-Organizing Maps.* Heidelberg: Springer-Verlag, 2nd edition.

Lim, A. 2001. Distributed services for information dissemination in self-organizing sensor networks for real-time systems with adaptive reconfiguration. *Journal of Franklin Institute* 338:707–727.