Towards a Human-Aware Operating System

Nicola Bicocchi
Dipartimento di Ingegneria Enzo Ferrari
Universita di Modena e Reggio Emilia
nicola.bicocchi@unimore.it

Damiano Fontana
Dipartimento di Scienze e Metodi dell'Ingegneria
Universita di Modena e Reggio Emilia
damiano.fontana@unimore.it

Franco Zambonelli
Dipartimento di Scienze e Metodi dell'Ingegneria
Universita di Modena e Reggio Emilia
franco.zambonelli@unimore.it

ABSTRACT

Body-area networks have become a key scenario for pattern recognition technologies. Applications range from implicit human-machine interactions, to autonomous monitoring of user habits and activities. This paper presents a general framework that provide developers with tools to orchestrate the continuous process of collecting and classifying data streams. This can facilitate the development of human-aware applications, i.e., applications that can adapt to the context of their users. The framework supports service oriented, reconfigurable components and provides a solid background to put at joint work specification- and data-driven approaches. It also provides an innovative meta-classification scheme allowing developers to implement applications by editing a state automata. Experimental results suggest that the approach could be integrated in a number of applications for: *(i)* improving energy efficiency, *(ii)* improving classification accuracy and *(iii)* improving software engineering of aware systems.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*; D.2.2 [**Software Engineering**]: Design Tools and Techniques; F.1.1 [**Computation by Abstract Devices**]: Models of Computation

1. INTRODUCTION

The adoption of sensor networks, actuators and computational resources capable of interacting with people is transforming urban environments as well as domestic spaces. People are increasingly exposed to machines and applications capable of understanding their health parameters, behaviours, activities, daily routines, and adapting their behaviour accordingly [13, 4]. To most extents, applications are increasingly becoming "human aware", i.e., capable of recognizing user activities in order to adapt to them.

However, the design and deployment of such services in future urban scenarios is still not trivial. In fact, it shakes current approaches until their foundations rooted in top-down design, and calls fort suitable general-purpose infrastructures (let's say "human-aware operating systems") to effectively support applications in reaching awareness with a bottom-up, unsupervised, approach.

Designing with a top-down approach means that all the requirements of a software architecture have to be taken in account a priori; systems engineered in this way have a predictable and measurable behaviour but are not well suited to cope with dynamic execution contexts. Contrarily, bottom-up design delivers robust systems and can be fruitfully used in pervasive environments. However, predicting and controlling their behaviour *by design* is not an easy task. Human-aware and socially-aware applications call for a balanced trade off between the two approaches.

Indeed, awareness appears to be one of the key driver to guide this tradeoff. In fact, while top-down design is still required given our current technological level, the awareness of operating conditions could be used to provide systems with bottom-up features proved to be essential in dynamic, interconnected, heterogeneous environments. Awareness modules are called for knowledge collection, organisation, and reasoning. Given their conception, and their task of providing adaptation capabilities to top-down artefacts called to operate open-ended environments, awareness modules themselves are needed to be rooted around adaptation concepts. In this context, the contribution of this paper is twofold:

- it describes an awareness framework for knowledge collection using industrial-level tools. It is able to collect data from a number of different sources and classify them using general-purpose algorithms. The framework is rooted around the concept of dynamic reconfiguration; modules can be loaded, unloaded, reconfigured, and rewired at runtime.
- it investigates how state based automata could be used to drive module reconfiguration and rewiring within our framework. In particular, we describe a life logging application in which a meta-classification scheme is used to modulate the use of sensor and classifiers at runtime. This mechanism proved to be effective in: *(i)* improving energy efficiency, *(ii)* improving classification accuracy and *(iii)* improving software engineering of aware systems.

BODYNETS 2014, September 29-October 01, London, Great Britain

DOI 10.4108/icst.bodynets.2014.257047

The rest of this paper is organised as follows. Section 2 motivates the work. Section 3 presents the global architecture of the awareness framework we are developing, while a life logging application is described and evaluated in section Section 4. Finally, Section 5 discusses related work and Section 6 concludes the paper.

2. THE NEED FOR RECONFIGURABILITY

Smart phones are increasingly equipped with computational, connectivity and sensing capabilities endlessly feeding online social networks. At the same time, autonomous ICT devices equipped with complimentary capabilities are pervading our cities. Their sensing capabilities are provided by *(i)* sensors networks and smart objects following the Internet of Things paradigm; *(ii)* near-field tags and technologies (NFC, RFID and Bluetooth). Socio-technical entities (i.e., citizens and ICT devices) are continuously and possibly invisibly involved in highly distributed and participatory sensing campaigns.

Recent works show that it is possible to recognise personal and social behaviours by analysing users activities both in physical and digital domains. However, they stress the relevance of mapping the intrinsic dynamism of real-word with engineering mechanisms pointing out three main challenges: *(i)* different approaches and algorithms are needed to deal with the many facets of the real-world situations; *(ii)* for the same classification problem, it is still not feasible to deal with every possible operative context with a single technique; textit(iii) classification accuracy is inversely proportional to the number of treated classes [15] [7].

Thus, modern architectures stemming from these requirements should exhibit a certain degree dynamism and flexibility and make applications able to autonomously interoperate and reconfigure. Service oriented and dynamically reconfigurable components have been proposed [11]. In particular they can be deployed and removed at runtime. These features allow to select among different components depending on the situation. For example, given a specific classification task, it is possible to select a highly precise and computationally expensive algorithm or a less precise one considering the device. Furthermore, reconfigurable components can transparently modify their internal parameters. For example, classifiers can analyse temporal windows of different sized based on the availably of computational resources, energy boundaries, or environmental conditions.

Despite reconfigurable components could provide a higher degree of flexibility to applications, the problem of driving reconfiguration processes is still open. In fact, every system capable of changing its internal structure or parameters must take decisions according to its environment. These decisions could be taken on different basis and could be the output of different reasoning processes. We decided to start driving internal reconfigurations with state-based automata because of their simplicity and generality. Each status is associated with a set of sensors specifically configured, while transitions and consequent reconfigurations are triggered by specific conditions. Moreover, such models can be monitored, enriched, updated at run time, and even shared among different applications and enriched over time by a community of developers. Engineers, in this way, could put at work a number of classifiers agnostic about the domain of the problem and dynamically reconfigure them using predefined set of rules.

On the opposite side, data-driven techniques usually produce numerical models that cannot be easily understood. Thus, designers willing to use these techniques for driving reconfigurable systems cannot understand the strategies automatically learned from data.

Summarising, the key idea behind this work consists in using service oriented, reconfigurable components to fill the gap between specification- and data-driven models. In fact, while data-driven models can be used to classify data streams, specification-driven models, such as automata, can drive the reconfiguration of the architecture. This meta-classifcation scheme leads to three main advantages:

- *Improve energy efficiency.* For each specific situation the less energy-demanding sensors and classifiers could be used. For example, it is possible to roughly recognise the vehicle used by a user with either GPS or accelerometer or microphone. An energy constrained system could constantly monitor its energy consumption and select the most appropriate trade-off. Furthermore, classification algorithms could be tuned to be less precise in favour of a minor computational complexity.
- *Improve classification accuracy.* Instead of having a single classifier looking for a wide spectrum of classes, several classifiers could be used to recognise only specific situations. These could be activated on-demand by taking in consideration the current context awareness of the framework (e.g. the vehicle classifier could be activate when the speed overcomes a certain threshold). Simpler classifiers are less prone to over fitting, are thus more general and could be used without expensive re trainings on other problems. Furthermore, as the number of situations (i.e., classes) increases combining simpler classifiers often gives better accuracy. The drawback of having several classifiers instead of a single one is in that possibly conflicting labels needing to be filtered/harmonised. Nevertheless, using specification-driven models to keep track of the current situation solves the problem and guarantees that labels produced by active classifiers are meaningful for that situation.
- *Improve software engineering.* Organising the framework around the idea of reconfigurable components (i.e., sensors, classifiers, meta classification schemes) leads to modularity of an innovative software ecosystem. On one side we will provide the community with a set of frequently used sensors and classifiers. On the other, users will be able to deploy their own both by using libraries already included in the framework or and by developing their own algorithms.

3. ARCHITECTURE

This section details an awareness architecture supporting both individual and collective awareness incorporating ideas described in Section 2. Specifically, its goal is to provide

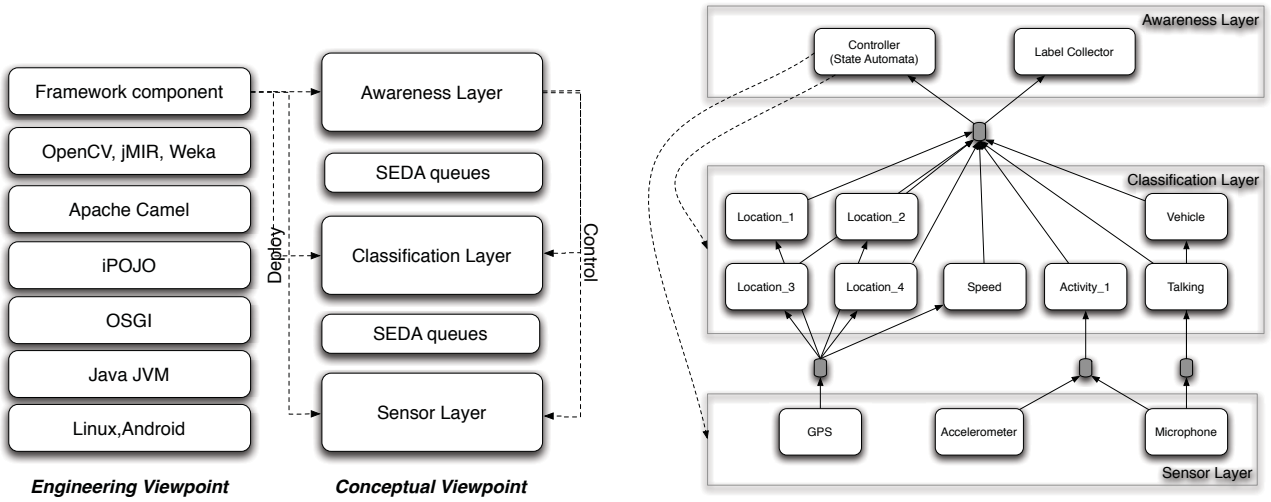

Figure 1: The framework architecture seen from both conceptual and engineering viewpoints. From the conceptual viewpoint, it is structured around three layers, namely *sensor*, *classifier* and *awareness* layer. From an engineering viewpoint, the architecture is implemented on the top of industrial-level Java technologies. Each module is actually an OSGi component enriched with iPOJO and Apache Camel functionalities (left). A mobile life logging application implemented using the framework (right).

general-purpose awareness that could be used as a starting point for many diverse applications. Developers are only required to select the needed modules, define the topology of data flows and specify their reconfiguration strategies as depicted in Figure 1.

The architecture is structured around three layers, namely *sensor*, *classifier* and *awareness* layer. Each layer can host multiple modules connected each other via application-definable topologies. The data flow from sensors (i.e., both hardware and software) through the whole architecture by means of in-memory queues enabling modules decoupling and many-to-many asynchronous communications. Each layer can host multiple modules (i.e, sensors, classifiers, awareness modules, queues).

The sensor layer hosts modules that are in charge of retrieving raw data from physical sensors and preprocess them. An example could be a module acquiring images from a camera and cropping and resizing them. Other examples could be modules acquiring facts from social networks, such as Twitter, Facebook or Foursquare. At the time of writing, we have already implemented modules for reading data from Android devices.

The classification layer hosts modules that consume data coming from the sensor layer and classify (i.e., generate semantically richer information) them. An example could be a module able to classify the activity performed by a user by processing accelerometer data. At the time of writing, we have implemented modules for classifying user activity, location, speed, vehicle used on the basis on common smartphone sensors. It is worth noting that our goal is to build a general-purpose awareness framework that could be used as common basis for both research and application development, not to solve every possible classification problem. Specific applications will need their own modules to be developed.

The awareness layer hosts modules consuming labels produced in the classification layer and feeding external applications with situational information. These modules might have different goals depending on the application. However, they could be divided into two main classes.

The former comprises modules delegated to sensor fusion processes. These modules receive labels, eventually conflicting, coming from multiple classification modules and apply algorithms to achieve higher semantical levels. For example, commonsense knowledge has been recently proposed [3] and could be integrated at this level.

The latter, instead, is related with the capability of the framework of monitoring and controlling itself. In a sense, the awareness layer could be the key of building a *self-aware* awareness module. For example, it would be possible to integrate within this level modules observing the internal status of the framework and activating different classifiers and sensors depending on operating conditions. This capability could be used to achieve both improved classification accuracies and reduced power consumption levels by continuously selecting to most suitable classifiers and sensors.

The strategies used to select sensors and classifiers might vary depending on the application. However, general purpose strategies could be identified. For example, in Section 4 we propose a meta-classification scheme based on a simple automata. Each state in associated to a set of active sensors and transitions are triggered by their labels. We believe that, a large numbers of real-world problems could be tackled with a relative small number of different automata.

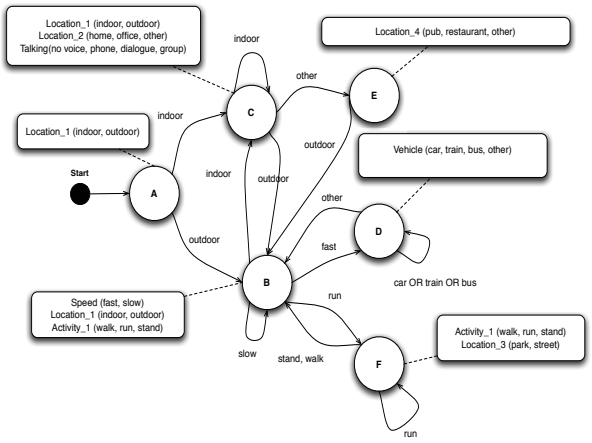

Figure 2: A self-aware life logging application collecting data about the life of the user such as activity, kind of location, vehicles used and people talking around.

From an engineering viewpoint, the architecture is implemented on the top of industrial-level Java technologies. Each module is actually an OSGi component able to meet the requirements mentioned in Section 2 [2].

On top of OSGi, we have an iPOJO layer. iPOJO is a container-based framework handling the lifecycle of *Plain Old Java Objects (POJOs)* and supporting management facilities like dynamic dependency handling, component reconfiguration, component factory, and introspection. Moreover, the iPOJO container is easily extensible and allows pluggable handlers, typically for the management of non-functional aspects.

On top of the iPOJO framework we build the support for the staged and layered architecture by making use of Apache Camel. This framework provides components with the capability of asynchronously processing data streams and communicate through in-memory queues. These queues allow modules belonging to different layers to continuously communicate each other with minimum hardware requirements. Considering that pattern classification and analysis has a central role in situation awareness, we wrapped well-know data manipulation libraries within the framework such as Weka and jMIR.

4. LIFE LOGGING, A CASE STUDY

We started experimenting with state-based automata because of their simplicity and generality. A number of real-world problems can, in fact, be described using this approach. Furthermore, automata are human understandable. Data-driven techniques usually produce numerical models that cannot be easily understood by a human. Thus, designers willing to use these techniques for driving reconfigurable systems cannot understand the strategies learned from raw data. Using automata, it is possible to drive the reconfiguration process using simple and clear schemas that could be continuously monitored and updated, even at run time.

To demonstrate that this approach could be successful we describe a self-aware life logging smartphone application. In particular it collects her activity, kind of location, vehicles used and people talking around. The system has the goal to collect data from various sensors, classify them and define the situation the user is immersed in. However, instead go using a single and complex classifier, developers can use a number of simpler and more specific classifiers. These modules can be enabled, disabled, wired and rewired in a dynamic way by making use of their output to navigate a state automata. For example, the automata shown in Figure 2 describes all the statuses used in this application. Each status has a name and is associated with a set of classifiers that have to be active and a set of possible transitions. Each time the output of an active classifier changes, a reconfiguration is applied and a new status reached. In this way, the overall problem of situation recognition is modularised in a way similar we believe our brain works. Each node embeds the knowledge acquired by the former and activates more specific classifiers to collect further details.

This reconfiguration strategy improves *(i)* energy efficiency because costly component in terms of energy consumption (e.g. audio classifiers) are activated on-demand; *(ii)* classification accuracy because classifiers work in optimal conditions (e.g. the topic classifier is activated only when a dialog is detected). Furthermore, *(iii)* software engineering is greatly simplified. One can simply draw an automata, associate to each status sensors and classifiers and deploy. As soon as the repository of available modules will be enough populated and graphical tools will be ready, the development of the awareness part of a number of different applications might become a trivial task.

4.1 Experimental Evaluation

We evaluated how an automata-driven meta-classification scheme impacts on both classification accuracy and energy consumption. More specifically, we used the reconfiguration capabilities of the framework to switch sensors on and off when needed. Each state is associate to a set of active sensors producing a set of labels that could be eventually used to trigger transitions. For example, state C activates both the location and activity sensors but triggers transitions to itself on state B using only location labels.

This process has two main advantages on the case in which all the sensors are kept on all the time. First, it is clear that power consumption could be significantly reduced by minimising the use of energy-hungry components. For example, when the system reaches state D, the only microphone (instead of GPS) is used to perceive changes. Second, classification accuracy usually improves because out-of-context classifiers do not produce misleading labels. For example, labels produced by the vehicle classifier are avoided when the user is located indoor without the use of any filtering technique.

We analysed data collected with a smartphone during a typical commuting working day. Our goal is to determine whether the user is at home, at office, which activity is performing and, in case she is moving, which vehicle is using.

The dataset has been collected using an Android Galaxy

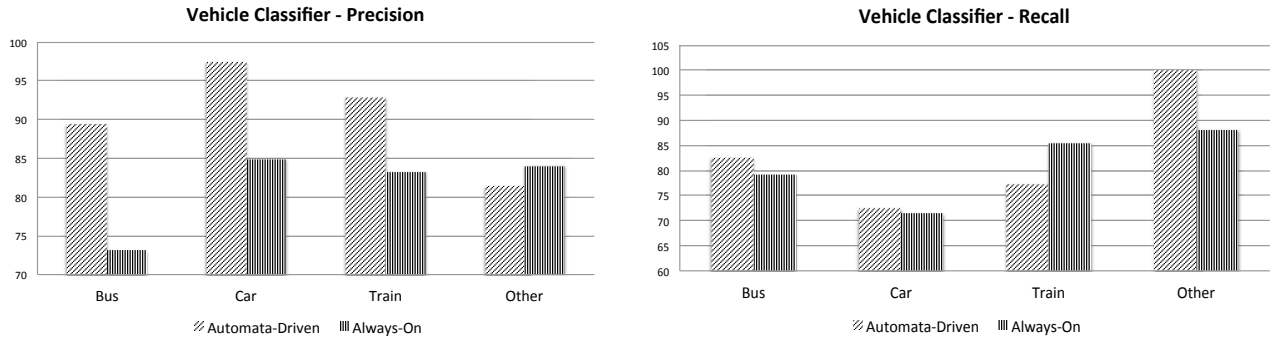

Figure 3: Precision and recall of vehicle classifier obtained by classifying a dataset with or without automata-driven reconfiguration.

Note II smartphone running a modified version of Funf [1]. Ground truth has been manually annotated. We defined a situation $s = \{activity, location, speed, vehicle\}$. Each field of the tuple can assume specific values. In particular $activity = \{walk, run, stand, sit\}$, $location = \{indoor, outdoor\}$, $speed = \{slow, fast\}$, $vehicle = \{car, bus, train, other\}$. Each field of the tuple is managed by a specific classifier (see Figure 1-right). In particular: *(i)* an activity classifier using accelerometer and microphone data; *(ii)* a location classifier recognising indoor and outdoor environments using GPS data; *(iii)* a speed classifier recognising fast and slow movements using GPS data; and *(iv)* a vehicle classifier using microphone data.

The activity classifier has a discriminative core based on SVM and uses a 64 dimensions Simple Magnitude Spectrum feature vector and its maximum value. It has been trained with accelerometer data sampled at 16 Hz. Feature vectors have been extracted using the jMIR workbench. The speed classifier computes the average speed over a sliding window of 10 GPS samples; while the indoor-outdoor classifier detect an indoor environment when there is a lack of data in a sliding window of 15 seconds. Finally, the vehicle classifier is based on SVMs and two feature vectors: *(i)* a 13 dimension Mel-Frequency Cepstral Coefficient (MFCC) feature vector, and *(ii)* a 10 dimension Linear Prediction Coeffecients feature vector. Data has been collected at 44.100Hz, 16bit, mono and divided in windows of 4 seconds long. Feature vectors are computed for each window using jMIR. We recorded 7200 seconds of GPS, accelerometer and microphone data streams collected by three different users. The dataset has been divided into two equal non-overlapping sets, one used as training, the other as test set.

Results, summarised in Figure 3, show that classification precision for the vehicle classifier increases around 10% while recall does not vary significantly. The improvement derives from the fact that automata's states gives memory to the system. Thus, all the errors produced by sensors working without context (e.g., a vehicle classifier working indoor or while users are walking in a busy street) are avoided. We believe that results could improve by analysing datasets collected from more sensors.

Furthermore, these results have been achieved with a substantial reduction of the energy required. Table 4 summarises the results. For each status, are reported both the sensors active and the percent of the execution time on the overall classification process. The GPS is active around 46% of time, the microphone 54% and the accelerometer 6%. It is clear from these data that the overall power consumption is around half the case with all sensors always-on. To give some additional details, let's consider a smartphone with an average 3100 maH battery and the following power consumption levels: CPU (50%) 100ma, accelerometer (16Hz) 80ma, GPS 140ma. With this common setup, an awareness module with all the sensors on would last around 10hours, while the presented framework around 20hours.

Exp-State	Active Sensors	Time (%)
Smartphone-A	GPS	3%
Smartphone-B	GPS	24%
Smartphone-C	GPS, Accelerometer	6%
Smartphone-D	Microphone	54%
Smartphone-E	GPS	3%
Smartphone-F	GPS	10%

Figure 4: Power consumption report: for each status, are shown both the sensors active and the percent of the execution time on the overall classification process.

5. RELATED WORK

In the last years, researchers prototyped sensing systems able to acquire detailed situational information from data streams [9, 8] using both specification-driven (e.g., logic ontologies, logic programming, fuzzy logic) or data-driven approaches (e.g., support vector machine, decision trees, neural networks). The most prominent works have been surveyed in [15] and [7].

However, the majority of these systems lacks in generality and addresses specific recognition problems, by making use of a pre-defined set of sensors [10]. Few of them tried to make use of both the approaches designing frameworks that are resource efficient and robust in a large plethora of situations. For example, in [11] authors make use of processing pipelines on various sensors (i.e., GPS, microphone, accelerometer) to show how processing pipelines and dynamic

reconfiguration could be used in continuous sensing. The framework doesn't focus on a generic approach enabling re-configuration and runtime adaptation. Furthermore, it doesn't use any specification-driven approach.

In [12], [5] and [14] authors propose to optimise the sensing process in terms of power saving. In particular, [12] exploits a specification-driven reasoning technique to learn relationship among context attributes to optimise the internal logic of an awareness framework. However, like previous works, these frameworks focus only the optimisation of energy consumption in continuous sensing. Kawsar et. al. [6] demonstrates that our preliminary approach of modelling human activities and contexts with a set of states glued by a set of transitions, i.e. a state based automata, can successfully describe a number of real world scenarios and drive the reconfiguration process.

Our framework is a first attempt to integrate both approaches. Its self-aware architecture makes it is able to implement all the strategies proposed in the previous works and to deal with complex scenarios requiring flexibility and adaptability as foundational basis.

6. CONCLUSION

We proposed a framework suitable for supporting general human-aware pervasive scenarios. It has been conceived around the concept on reconfiguration and built using well-established technologies. Its modular and portable architecture makes it suitable for a number of applications in which awareness either individual or collective could be used for triggering adaptation. Furthermore, an innovative meta-classification scheme based on sensor reconfiguration has been experimented. Results showed that the approach could be useful for: *(i)* improving energy efficiency, *(ii)* improving classification accuracy and *(iii)* improving software engineering of aware systems.

Acknowledgments

Work supported by the ASCENS project (EU FP7-FET, Contract No. 257414).

7. REFERENCES

[1] N. Aharony, W. Pan, C. Ip, I. Khayal, and A. Pentland. Social fmri: Investigating and shaping social mechanisms in the real world. *Pervasive and Mobile Computing*, 7(6):643–659, 2011.

[2] P. Bellavista, A. Corradi, D. Fontana, and S. Monti. Off-the-shelf ready to go middleware for self-reconfiguring and self- optimizing ubiquitous computing applications. In *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*, 2011.

[3] N. Bicocchi, M. Lasagni, and F. Zambonelli. Bridging vision and commonsense for multimodal situation recognition in pervasive systems. In *International Conference on Pervasive Computing and Communications*, Lugano, Switzerland, 2012.

[4] G. Castelli, A. Rosi, M. Mamei, and F. Zambonelli. A simple model and infrastructure for context-aware browsing of the world. In *IEEE International Conference on Pervasive Computing and Communications*, pages 229–238, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[5] S. Kang, J. Lee, H. Jang, Y. Lee, S. Park, and J. Song. A scalable and energy-efficient context monitoring framework for mobile personal sensor networks. *Mobile Computing, IEEE Transactions on*, 9(5):686–702, 2010.

[6] F. Kawsar, G. Kortuem, and B. Altakrouri. Designing pervasive interactions for ambient guidance with situated flows. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 3, pages 371–375, 2010.

[7] W. Z. Khan, Y. Xiang, M. Y. Aalsalem, and Q. Arshad. Mobile phone sensing systems: A survey. *IEEE Communication Survey and Tutorials*, 15:402 – 427, 2013.

[8] J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Activity recognition using cell phone accelerometers. *SIGKDD Explor. Newsl.*, 12(2):74–82, Mar. 2011.

[9] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. Energy-accuracy trade-off for continuous mobile device location. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 285–298, New York, NY, USA, 2010. ACM.

[10] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile phones. In *Proceedings of the 7th international conference on Mobile systems, applications, and services,*, pages 165–178, 2009.

[11] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. The jigsaw continuous sensing engine for mobile phone applications. *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 71–84, 2010.

[12] S. Nath. Ace: exploiting correlation for energy-efficient and continuous context sensing. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, MobiSys '12, pages 29–42, New York, NY, USA, 2012. ACM.

[13] N. Roy, T. Gu, and S. K. Das. Supporting pervasive computing applications with active context fusion and semantic context delivery. *Pervasive and Mobile Computing*, 6(1):21 – 42, 2010.

[14] M. Schirmer and H. Höpfner. Senst*: Approaches for reducing the energy consumption of smartphone-based context recognition. In M. Beigl, H. Christiansen, T. Roth-Berghofer, A. Kofod-Petersen, K. Coventry, and H. Schmidtke, editors, *Modeling and Using Context*, volume 6967 of *Lecture Notes in Computer Science*, pages 250–263. Springer Berlin Heidelberg, 2011.

[15] J. Ye, S. Dobson, and S. McKeever. Situation identification techniques in pervasive computing: A review. *Pervasive and Mobile Computing*, 8:33–66, 2012.