Revised paper – Dec 1, 2010.

MobiCon: Mobile Context Monitoring Platform for Sensor-Rich Dynamic Environments

[1]Seungwoo Kang, [2]S.S. Iyengar, [1]Youngki Lee, [1]Chulhong Min, [1]Younghyun Ju, [1]Taiwoo Park, [1]Jinwon Lee, [3]Yunseok Rhee, [1]Junehwa Song.
[1]Department of Computer Science, Korean Advanced Institute of Science and Technology (KAIST), South Korea.
[2]Department of Computer Science, Louisiana State University, Baton Rouge, USA.
[3] Electronics & Information Eng., Hankuk University of Foreign Studies, South Korea.

ABSTRACT

We present a new mobile context monitoring platform to support emerging pervasive applications in a Personal Area Network (PAN)-scale dynamic mobile computing environment. Composed of a mobile device and many wearable and space-embedded sensors, the PAN-scale computing environment will constitute an important part of future pervasive spaces technology. The monitoring platform presented in this paper supports a number of context-aware applications to simultaneously run and share highly scarce and dynamic resources. Our new system develops a novel translation-based approach for efficient and effective monitoring architecture for the environment. We implement and test a prototype system on multiple mobile devices with a diverse set of sensors. Example applications are also developed based on the implemented system design. Experimental results show that the system achieves a superior level of scalability and energy efficiency. The processing time can be reduced by orders of magnitude compared to what the present technology offers, and data transmission can be reduced by the order of 50 percent.

1. INTRODUCTION

Smart mobile devices will be the central gateway for personal services in the emerging pervasive environment (Figure 1). They will enable a lot of personal context-aware applications, forming a personal sensor network with a number of diverse sensor devices, placed over human body or in surrounding spaces. Diverse sensors act as the useful tool for the applications to acquire users' contexts[1], i.e., current status of an individual or surrounding situation that she/he faces into, without their intervention [42]. Table 1 summarizes example contexts, sensors, and related applications. For example, diverse physical contexts such as heart rate can be recognized from biomedical sensors such as ECG, GSR, and BVP sensors, and gait from accelerometers and gyroscopes. Also, environmental status can be obtained from light/ temperature/dust sensors, GPS, RFIDs in daily surrounding spaces [5][18][19][23][24] and the network of such sensors [26][27]. Diverse contexts enable mobile applications to proactively provide users with personal services customized to their situation. Such context-aware applications increasingly emerge in practice and in diverse research domains, e.g., healthcare [3][29], elderly support [25], dietary monitoring [2], daily life assistant [4], and sports training.

Proactively providing context-aware services often needs continuous monitoring of users' context. The context monitoring is a process of continuous detection of context changes. It requires continuously collecting data from diverse sensors, processing the data to derive context, and keeping track of changes of users' context. This semantics is different from conventional context recognition, which identifies the current context. Once a change is identified, it is not necessary to recognize the context redundantly as long as it remains unchanged. The context monitoring often involves multi-step complex operations such as feature extraction and context recognition distributed across the mobile and sensor devices at the same time. However, it is quite difficult for individual applications to perform complicated context monitoring process on their own, especially over the shared devices with highly limited resources. To effectively support context monitoring, a common monitoring platform is essential.

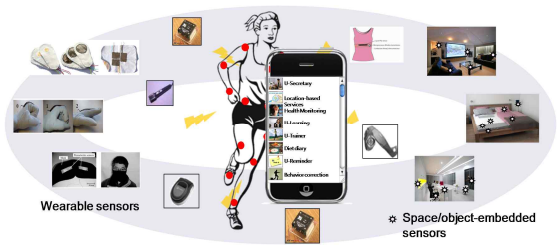


Figure 1. PAN-scale Sensor-rich Mobile Environment

[1] While one of important contextual information is profile-based context such as age, gender, profession, contacts, and SNS buddy lists which is rather static, this paper focuses on sensor-enabled context which is more dynamic status information derived from sensing data.

Table 1. Example contexts and applications

Category	Context	Application	Sensor
Health	Fall	Elderly care [25]	Accelerometer Gyroscope Sound
	Motion, Gait	Patient monitoring (e.g. Parkinson's disease) [28]	Accelerometer Gyroscope
	Heart condition (IBI, heart rate, …)	Heart monitor, Jogging support	ECG, BVP, SpO2
	Calorie	Activity-based calorie monitor [29]	Accelerometer
Activity and Status	Affective Status	SympaThings [18]	BVP, GSR
	Activity	Activity-based calorie monitor [29] Activity-based service initiation	Accelerometer Gyroscope Camera RFID (object-attached)
	Gesture	User interaction [31]	Accelerometer Gyroscope Camera
	Sleep pattern	Quality sleep monitor	Accelerometer, Sound
Location	Outdoor Location	Navigation Traffic monitoring Microblogging Mobile advertisement Personal logging	GPS, Accelerometer
	Indoor Location		WiFi infrared Ultrasound
Place	Mood Crowdedness Noise Vacancy Illuminance	Recommendation Social Networking Party thermometer	Sound sensor Camera Illuminometer GPS/WiFi
Environment	Weather	Weather monitor	Temperature Humidity Wind speed
	Pollution	City Pollution monitor	CO2, O3, Dust, S
Sports Training	Golf Swing	Virtual golf trainer	Accelerometer
	Swim posture	Wearable swim assistant [30]	Accelerometer
	Treadmill	SwanBoat [21]	Proximity sensor Accelerometer
Companion	Number Relationship	Advertisement Groupware	Sound Proximity

Challenges and Existing Approaches

Enabling mobile context monitoring involves complex, multilateral challenges spanning over different research and technical issues. Applications require different types of contexts in different degrees of awareness and accuracy. Users have different requirements and preferences for the services as well as privacy concern. Efficient resource utilization and management also becomes more important issue for continuous context monitoring. To enable diverse context-aware application with ease and efficiency, in particular, infrastructural supports including a platform are required. Such challenges have been addressed in diverse research domains evolving independently with different aspects and emphasis. We introduce such research efforts in four major areas.

Artificial intelligence and machine learning. Many research efforts have been done to recognize user contexts automatically, especially focusing on recognition accuracy and complex context extraction from sophisticated sensors such as camera, e.g., arm tracking and face gesture detection. They have addressed a variety of challenges across multi-stages in context recognition, e.g., pre-processing, feature extraction, segmentation, and recognition algorithms. They try to extract the most effective features for accurate recognition (e.g., MFCC for audio [49], statistical or FFT features for activity [5]) from raw sensing data (e.g., camera, microphone, biomedical and motion sensors). From continuous feature data, data sections corresponding to valid context values are automatically segmented without user intervention (e.g., probability-based segmentation [50]). Most importantly, diverse context recognition algorithms have been studied to accurately classify the segmented features into defined context vocabularies. (e.g., HMM for speech and gesture recognition [50], Decision tree for activity recognition [5][18], SVM for face recognition [51]). These algorithms are very powerful in handling uncertainty of contexts as most of them are based on probabilistic inference and reasoning. However, it is difficult to directly apply the research work to a real mobile environment. In general, context monitoring would be performed in resource-constrained sensor-enabled mobile platform, especially with limited battery and computing power. Hence, the continuous processing of complex AI and ML algorithm-based context monitoring imposes heavy workload on the mobile platform, thereby limiting their practicality.

Mobile and sensor systems. To enable context monitoring, it is essential to manage and efficiently utilize highly limited resources of mobile and sensor devices. There has been previous research which mainly addressed the challenges of resource limitation on mobile and sensor systems. They mainly include (1) OS and system middleware that provide common system functionality to execute mobile and sensor applications in a resource-aware/adaptive manner (e.g., ECOSystem [44] and Odyssey [41][54] for mobile systems, Pixie [39] and Eon [32] for sensor systems), (2) component techniques for resource efficiency that optimize the resource utilization in executing application tasks, especially energy consumption [33][34][35][37][38], (3) distributed resource management to manage resource use across multiple devices or applications (e.g., load balancing in sensor network [45], adaptive node selection in dynamic networks [55]). Unfortunately, most of existing works in this area did not focus on context monitoring applications, and hence, their architectures and mechanisms are not fully optimized for context monitoring applications.

Context-aware applications. A variety of context-aware applications and application-specific systems have been developed, e.g., healthcare and medical applications [3] reminder applications [4] and activity recognition [5]. Each system mainly utilizes an application-specific context such as location, activity or biomedical information. To some extent, they have enabled useful application functionality in various domains. However, they are limited in providing a general platform to support multiple applications which

utilize diverse sensors and associated contexts. In addition, the careful consideration over resource limitation has not been made since the platform is not designed for the shared use of multiple applications.

Context middleware. Some existing projects have proposed middleware to support context-aware applications. Their aim is to hide the complicated issues related to context inference and facilitate applications to obtain contexts of interest at a time. However, they are limited in terms of continuous monitoring in sensor-rich, resource-limited environments. Most of initial context middleware is designed to run in a server environment [6][7][8][9]. Thus, they do not deal with the issues regarding severe resource constraints of the mobile and sensor environment. Some context-aware middleware targets mobile devices [10][11] but does not consider tens/hundreds of PAN sensors and the processing and power limitations of mobile devices and sensors.

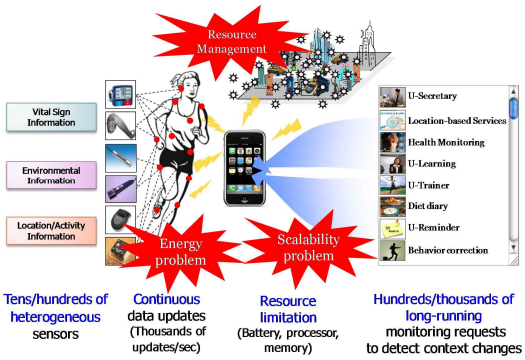


Figure 2. Challenges in Mobile Context Processing

Research Objective of MobiCon

We aim at implementing a practical context monitoring system for context-aware applications working over sensor-rich and resource-limited mobile environments (Figure 2). To develop such a system, it is critical to consider unique challenges arisen in the environment, which widely spans above-mentioned research areas. First, the system should be able to continuously run complicated multi-step computation over mobile and sensor devices for accurate context recognition. The computation takes significant amount of computing resources; it often involves high-rate data sampling from multiple sensors, continuous feature extraction and context recognition over massive sensor data, and transmission of intermediate results among devices. Due to the scarce battery and computing resources, the continuous and long-running operation is highly challenging. Second, the system should concurrently support a number of such complicated requests from multiple applications as a shared underlying platform. This makes the resource limitation problem a lot more serious. Also, the system should effectively resolve the potential conflicts among multiple applications in their resource utilization. Finally, dynamic availability of wearable or space-embedded sensors and their varying resource status impose a new challenge to support continuous monitoring for seamless services.

To show the challenges with a quantitative data, we study the resource limitation considering the off-the-shelf sensor devices and example context-aware applications. Table 2 shows the resource availability of example sensor motes. In practice, available computing resources such as CPU and memory of sensor devices are frequently less than the capacity required for a single context monitoring. For example, a MicaZ mote has 8MHz CPU, 4KB RAM, etc. It is even incapable to run a light FFT library, kiss_fft [43] which is frequently used to monitor activity. In addition, highly scarce battery capacity could compromise the continuous monitoring, which results in early shutdown of relevant applications. Table 3 shows average energy consumption of example tasks performed on a Knode mote. Given the battery capacity of Knode, i.e., 250mAh, the achievable lifetime is about 21.5 hours to execute tasks for fall detection and heart beat monitor (the third case in Table 3). Concurrent execution of additional tasks aggravates energy shortage even more severe. In [28], an energy deficiency has also been reported with a SHIMMER sensor; the worst-case lifetime was 9.2 hours to perform a range of tasks for motion analysis including continuous sampling and logging of accelerometer and gyroscope data, maintaining time sync, and raw sample transmission.

We develop MobiCon, a novel mobile context monitoring framework, which effectively addresses the new challenges arisen in the sensor-rich mobile environments to support a number of context-aware applications over a personal sensor network. Although the above-mentioned research efforts in different directions provide basic techniques to enable context monitoring, careful reconsideration on different fields are essential to implement a practical context monitoring system addressing the new inter-disciplinary challenges. This is mainly because each research direction focuses on addressing its own challenges. Our research can be considered as initial step to bridge the gap among different research fields to enable context monitoring. Furthermore, we propose a novel solution approach to effectively address the issues to overcome the limitations of existing works to fully support context monitoring in the sensor-rich mobile environment.

Table 2. Resource availability of example sensor motes
(Knode is a custom-designed sensor mote as a MicaZ variant.)

	MCU	Comm.	Flash memory	Onboard sensor	Optional sensor	Battery
SHIMMER [28]	TI MSP430	CC2420 (802.15.4, 2.4GHz, 256Kbps)	3GB	3-axis accelerometer	Gyroscope, ECG, EMG, etc.	25o mAh Li-polymer rechargeable
Knode	Atmega128 (8MHz CPU) (4KB RAM)		1MB	3-axis accelerometer 2-axis gyroscope Light	Gyroscope (to cover 3D), SpO_2	250mAh Li-polymer rechargeable
USS2400 (MicaZ clone)			None	None	2-axis accelerometer, thermometer, etc.	2 x 1200mAh alkaline

Table 3. Energy consumption of applications on a watch-type Knode

Application	Tasks	Avg. energy consumption (mJ/s)
Fall detection 1	**Sensing**: 30Hz from 3D accelerometers **Feature extraction**: 'norm' **Transmission:** 6Hz to a mobile device	3.6
Fall detection 2	**Sensing**: 30Hz from 3D accelerometers and 2D gyroscopes **Feature extraction**: 'FFT' with window size of 64 and sliding window of 32 **Transmission:** 1.07Hz to a mobile device	13.1
Fall detection 1 + Heart beat monitor	Tasks for Fall detection 1 + **Sensing**: 60Hz from SpO_2 sensor **Feature extraction**: peak detection and count **Transmission**: 1 Hz to a mobile device	33.8
Base energy consumption (no task)		8.8

An overall structure of MobiCon is given in the following section. The subsequent sections will elaborate on the components of MobiCon and how they are integrated. Please note that many details of the components and techniques are available in [15, 16, 18, 19, 20, 21, 22, 52, 53].

2. FRAMEWORK DEVELOPMENT

MobiCon is a middleware framework mediating context-aware applications and a personal sensor network (Figure 3). MobiCon provides Application Programming Interfaces (APIs) as shown in Table 4 and a runtime environment for applications. Multiple applications that require context monitoring can be developed through the APIs and can run on top of MobiCon concurrently. MobiCon acts as a shell above the personal area network and provides a neat interface for the users for receiving and processing sensor data and controlling the sensors.

MobiCon consists of five components: the *Context Processor*, the *Sensor Manager*, the *Resource Coordinator*, the *Application Broker*, and the *Sensor Broker*. Applications initiate context monitoring by registering *context monitoring queries* to MobiCon through the Application Broker. Then, the Context Processor performs context monitoring by evaluating the queries over data delivered by the Sensor Broker; monitoring results are then forwarded to applications. In this process, the Sensor Manager finds a minimal set of sensors that is necessary to evaluate all registered queries, and forces unnecessary sensors to stop transmitting data to MobiCon. In addition, the Resource Coordinator mediates potential conflicts in the use of resource-scarce sensor nodes when handling multiple requests from concurrent applications. Also, it supports adaptation to the dynamic sensor join/leave.

We have implemented the MobiCon system architecture as a prototype system, carefully applying the translation-based approach. The current prototype is implemented in C++ based on a Linux operating system.

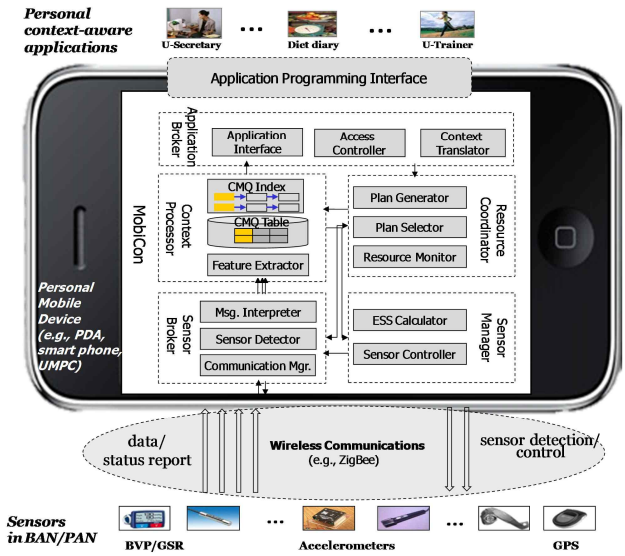


Figure 3. Architecture of MobiCon

● Prototype Hardware

Deploying MobiCon requires two important hardware sets: mobile devices and sensors. Figure 4 shows a snapshot of currently used hardware including the two mobile devices and sensors. We have deployed the MobiCon prototype and its applications on two different mobile devices: (1) an Ultra Mobile PC (UMPC), SONY VAIO UX27LN with Intel® U1500 1.33 GHz CPU and 1GB RAM (See Figure 4 (e)), and (2) a custom-designed wearable device with Marvell PXA270 processor and 128MB RAM (Figure 4 (c)).

We have incorporated as many sensors that a person can carry to increase the coverage and accuracy of context monitoring of MobiCon. We prefer small-size controllable sensors with processing and wireless communication capabilities appropriate for mobile environments. Such sensors can be deployed in a wearable or a carry-able form and adopt the sensor control mechanism of MobiCon easily.

Considering this, we mainly use two types of MicaZ clone sensor nodes; USS-2400 and Knode. First, we have eight of USS-2400 sensor nodes, i.e., four 2-axis accelerometers, two light, and two temperature/humidity sensors as shown in Figure 4 (h). Second, we designed and utilized Knode to support diverse sensing modalities in a single mote. The Knode is equipped with Atmega 128L MCU, 1G Flash memory, CC2420 RF transceiver supporting 2.4GHz band ZigBee protocol, and TinyOS as an operating system. It basically includes 3-axis accelerometer and 2-axis gyroscope, and support extensions. Figure 4 (d) shows the current prototype which incorporates four basic Knodes and one extended Knode with SpO_2 sensor module.

We incorporated several additional sensors to provide important context types not supported by the MicaZ clone sensor nodes. First, we use three biomedical sensors, an Electrocardiogram (ECG) sensor (Figure 4 (g)), a Blood Volume Pulse (BVP) sensor and a Galvanic Skin Response (GSR) sensor (Figure 4 (f)). We also incorporate a Bluetooth-enabled GPS sensor (Figure 4 (i)) to position outdoor location.

We developed wearable sensor devices with different form factors, Sensor-Bracelet (Figure 4 (a)) and U-Jacket (Figure 4 (b)). Sensor-Bracelet is equipped with the basic Knode. U-Jacket incorporates the BVP and GSR sensors.

Figure 4. H/W devices

3. Translation-based Approach for Mobile Context Monitoring

In this paper, we discuss the issues and approaches to build an efficient mobile context monitoring platform for PAN-scale sensor-rich dynamic environment.

Figure 5 shows the flow of conventional context monitoring process. The processing usually proceeds in one direction through a pipeline of several stages, i.e. preprocessing, feature extraction, and context recognition, which is fixed as defined by a programmer. It is beyond a single-step recognition task, and rather should continuously recognize a context of interest and detect changes in the contexts as a long-lasting task. In this approach, the change detection is always performed at the last stage. This model has been employed by individual applications in their own ways. It has usually supported a single or small number of monitoring requests using a small number of sensors under a rather general computing environment with sufficient resources.

Such a conventional approach forms a separate and heavy processing pipeline for each context. As a result, it is not efficient enough for supporting a number of concurrent monitoring requests on a mobile platform and sensors with highly constrained resources. In this approach, each application mostly uses resources in a greedy manner, and it worsens resource deficiency. With a limited view on system status, especially, it is not easy for individual applications to resolve resource contention with each other. Also, they could hardly achieve global efficiency and sharing in resource uses and computation on their own. Further, it is also hard for the applications to adapt themselves according to dynamic resource availability. In essence, the fixed processing defined beforehand by a programmer restricts diverse attempts to mediate between system and applications.

To address the limitation of the conventional approach, we propose a translation-based approach for a mobile context monitoring platform. In the approach, applications specify the monitoring requests of their interest in a high level language and submit them to the platform. Each request is translated to a lower-level representation by the system. Once translated, the lower-level representation gives the platform a chance to attain rather detailed understanding on application requirements as well as low-level status of associated sensor resources. In addition, detailed resource status can be dynamically analyzed considering the application requirements. With such understanding, the platform can support a feedback path between each application and its lower-level processing, and further extend the computational stages in the processing pipeline, significantly saving computational as well as energy overhead. Moreover, it can take an active role to orchestrate the application requests and system resources, and significantly improve overall performance of the system.

This translation-based approach primarily enables each application to concisely inform a system of its all requirements. The system can easily grasp a comprehensive understanding on each application, especially in terms of resource and computational demands. Holistically considering system status including resource availability and all running applications, furthermore, it can adaptively determine a globally optimal translation for each monitoring request. As a result, this translation-based approach enables a system-level orchestration of resource uses between competing applications over highly dynamic and resource-constrained environments. For instance, it can mediate the applications to achieve a system-wide goal such as energy balancing which could be hardly accomplished by each application-level strategies.

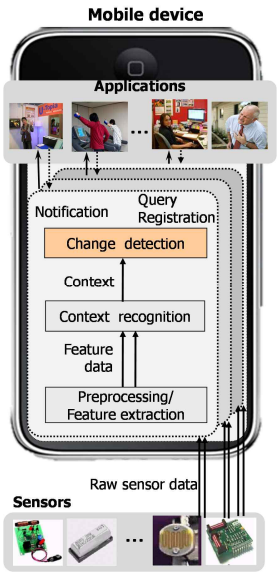


Figure 5. Conventional context monitoring process

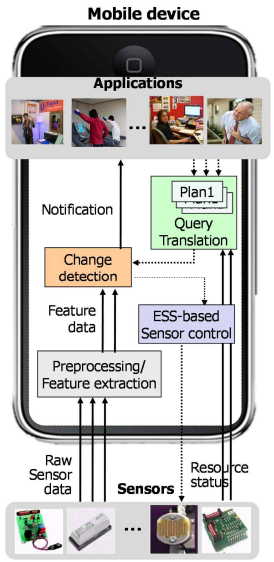


Figure 6. Translation-based approach to context monitoring problem

Figure 6 illustrates the flow of the translation-based context monitoring process presented in this paper. As shown in the figure, the system adaptively determines an appropriate resource use plan for each application's monitoring request according to dynamic resource availability. We note that the system prepares in advance several alternative plans for each monitoring context. Unlike conventional approaches, especially, it supports a system-level optimization in resource uses and computation. It could also reorganize processing stages so that change of context is identified efficiently at an early stage of the processing pipeline. This is contrary to the conventional way to detect the changes only after inferring them via an algorithm like decision tree logic. The costly operation such as decision tree logic can be avoided if the change of activity is detected earlier by feature value changes from accelerometers.

In terms of processing efficiency, the translation-based approach also gives a chance to develop an advanced acceleration method for context processing. Some dependency among seemingly un-related contexts is frequently exposed through the lower-level representation, and the related contexts are grouped together. Then, a shared operator can be composed, with which the grouped requests are evaluated by a single processing. The sharing can be achieved in several levels such a processing stage, feature extraction, and sensor reading. The feature level specification also allows taking advantage of continuity of contexts and the locality in consecutive sensing data streams. An incremental processing method is developed to accelerate the successive monitoring processes exploiting such locality in consecutive input data.

An efficient sensor control mechanism is further developed to enhance the energy efficiency of sensors and mobile devices. The key idea is that given a number of queries and sensors, only a subset of sensors is often sufficient to answer the queries. For example, a query with the context "studying in the library" would be answered without activating an accelerometer to recognize the user's activity if it is known that she is not in the library. The feature level specification of requests gives better chance to identify the sensors that are essentially required to process application's monitoring requests. A sensor control method can be developed to compute and activate only a reduced set of sensors, called Essential Sensor Set (ESS). Such a sensor control mechanism will reduce the amount of wireless communication between sensors and a mobile device, leading to energy savings.

Lastly, the translation-based approach relieves developers from devising their own context recognition methods which are mostly quite complicated and require considerable expertise. Also, it enables common context vocabularies to be effectively shared and thus a new personal context application to be easily developed in a fast time.

4. APPLICATION INTERFACES

A context monitoring platform should provide application developers with intuitive and generic context monitoring APIs. Through the APIs, applications can simply delegate complex context monitoring tasks to the platform and focus on application specific logics such as UI beyond the context monitoring. Applications do not need to specify which sensors to use, what types of data to collect, how often to collect, which feature extraction and classification modules to apply, and which computing resources to utilize to execute modules.

As a core of context monitoring API, MobiCon provides *Context Monitoring Query* (CMQ). CMQ is a declarative query language that intrusively supports rich semantics for monitoring a wide range of contexts while abstracting complicated resource details of devices. Also it is designed to catch the changes in users' context proactively. A CMQ specifies three conditions: *context*, *alarm*, and *duration* conditions. The alarm condition determines when MobiCon delivers an alarm event to applications. The duration condition specifies how long a registered CMQ should be evaluated. The following is an example CMQ.

```
CONTEXT (location == Library)
        AND (activity == Sleeping)
        AND (time == Evening)
ALARM F → T
DURATION 120 DAYS
```

CMQ is registered or deregistered by a set of APIs supported by MobiCon (Table 4).

Table 4. MobiCon API

Functionality	API List
Context monitoring-related APIs	registerCMQ(*CMQ_statement*)
	deregisterCMQ(*CMQ_ID*)
Query translation map-related APIs	createMAP(*[Parent_Map_ID]*)
	deleteMAP(*Map_ID*)
	browseMAP()

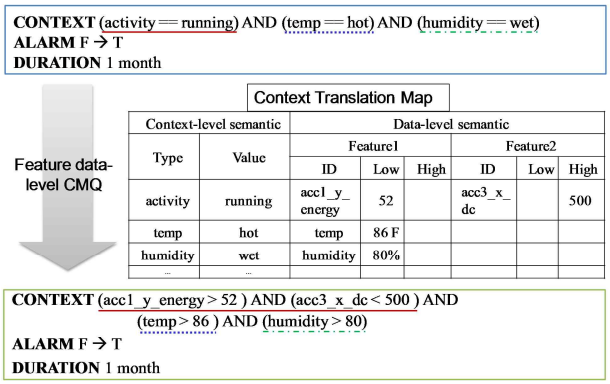


Figure 7. Example of context translation map and CMQ translation

- Bridging logical and physical resources: CMQ Translation

CMQ translation is a key to realize the translation-based approach. By bridging the gap between high-level (logical) application requests and low-level (physical) resource utilization, it acts as the critical base to enable efficient context monitoring and resource orchestration.

CMQ translation process converts CMQs specified in context-level semantics into range predicates over continuous feature data, their associated sensors, and corresponding resource demands. A context type specified in a registered CMQ is mapped to one or more features and their associated sensors. One or more features can be derived from a sensor. For example, DC and energy features are derived from an accelerometer [5]. Next, a context value is transformed to numerical value ranges for corresponding features. For example, (humidity == wet) can be mapped to (80% < humidity). Also, resource demands to compute feature data are specified. Feature computation can be performed on sensor devices or a mobile device. Accordingly, corresponding resource demands can be different. Resource demands are collected by offline profiling.

A *context translation map* is maintained to support the CMQ translation effectively. Figure 7 shows a simple case of CMQ translation based on an example context translation map. Note that the translation cost is negligible since the translation is a simple one-time operation performed for query registration.

5. RESOURCE COORDINATION

It is highly challenging to perform context monitoring with highly scarce and dynamic resources of sensor devices. Greedy and careless resource uses would significantly aggravate contentions among multiple applications. This can substantially reduce potential system capacity. Most of the devices have highly constrained resources, often less than the capacity required for a context monitoring. Moreover, availability of such devices is dynamically changing due to their wearable form and mobility of users. It is almost impossible for each application to address the challenges without system-level supports. An application itself has a limited view of the existence or resource uses of other concurrent applications, and cannot negotiate with the applications for coordinated resource utilization.

In many of existing mobile and sensor systems such as Pixie [39], Chameleon [40], and Odyssey [41], resource management approaches are so far mostly based on application-driven decision in resource allocation, i.e., passive resource use management. They expose APIs for resource allocation to applications. Applications determine the types and amounts of resources required to execute program codes, and explicitly request the resources through the provided APIs. For example, Pixie provides resource tickets, e.g., <Energy, 700mJ, 10sec>, and Chameleon provides systems calls such as set-speed() to control CPU speed directly. Then, a system simply allocates the requested resources if available. If not, applications would change their resource utilization according to predefined code blocks, e.g., by trading off data or functional fidelity. However, these approaches impose huge burden to programmers, and their flexibility cannot be fully utilized in practice since it is almost impossible for applications to estimate dynamic resource status and prepare well adapted code blocks for all the cases.

For effective resource coordination, we take an active resource use orchestration approach in which a system extensively explores alternatives for resource use [20]. The translation-based approach provides a key method to enable the active resource coordination in the system-level. We remind that each application submits a high-level context specification to the system, and also delegates to the system the decision how to process the context with what resources. That is, this approach decouples resource selection and binding from applications' logical resource demands. Then, the system actively finds the best combination of resources to process the context specifications on-the-fly under the current status of resources and applications.

The uniqueness of the proposed approach can be characterized as follows (Figure 8). First, MobiCon generates multiple alternative resource use plans to process a high-level context from applications. Such alternatives result from the diversity of semantic translation. A context can be derived by utilizing different processing methods.

For instance, when the context quality level required by an application is conditionally tolerable under a situation, a 'running' context is monitored with diverse methods, e.g., utilizing DC and energy features from acceleration data or statistical features from GPS location data.

Second, MobiCon dynamically selects a set of plans to execute in the runtime. The selection is performed in a holistic manner, reflecting resource availability, application requests, and a system-level policy. Such plan selection allows MobiCon to resolve contentions and maximizes sharing in resource use of multiple applications. At the same time, it best meets a system-wide policy for resource use, e.g., minimizing the total energy consumption.

Third, MobiCon continuously changes executed plans to adapt to dynamic system environment. A set of plans selected at one time may not persistently assure the best resource use since resource availability and application requests constantly change over time. Upon a change, MobiCon updates existing plans and reselects plans that result in resource use to still meet the policy under changed system conditions. Such flexibility and adaptation enable MobiCon to support multiple applications as long and balanced as possible in the dynamic environment.

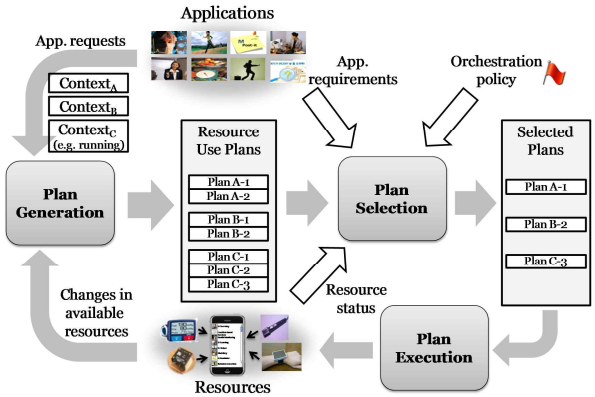


Figure 8. Active resource use orchestration approach

6. SHARED EVALUATION OF CONTEXT MONITORING REQUESTS

Context monitoring is a complicated processing composed of multi-step complex operations such as feature extraction and context recognition. Also, there are a large number of requests from multiple applications. Further, the requests should be continuously evaluated upon the arrival of each sensing data from a number of sensing data streams; any changes in the contexts of interest should be detected and notified in real-time. The processing should be done in a resource-limited mobile and sensor devices. It is essential to develop an efficient processing method to evaluate a number of requests continuously in such an environment.

There has been much effort to accelerate a single processing pipeline of context recognition consisting of complex feature

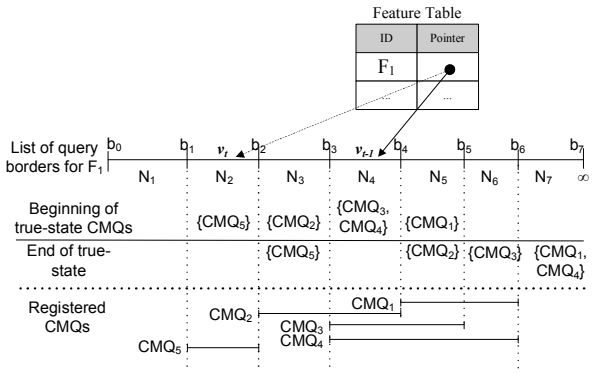


Figure 9 CMQ-Index: the index shows a list of query borders for feature F_1 and five CMQs.

extraction and pattern classification steps. First, feature reduction methods have been proposed to lessen the number of features which rapidly increases processing time, e.g., Sequential Forward and Backward Search (SFFS) and Linear Discriminant Analysis (Fisher LDA) [46]. Also, feature quantization was beneficial to reduce the computation cost of recognition algorithm by using a few discrete feature values, rather than continuous feature values [47]. Finally, recognition algorithm selection have been widely considered in that the processing time of non-stochastic algorithms (e.g., Decision Tree, SVM) is much lower than that of stochastic algorithms (e.g., Bayesian Network, Neural Network) [48]. Hence, non-stochastic algorithms were preferred if both types of algorithms show a similar recognition accuracy. While these techniques are complementary to our approach, our system is further specialized in accelerating multiple processing pipelines of context recognition.

To address the issue, a context monitoring platform can take advantage of shared evaluation of multiple application requests. The platform can look into the requests from different applications and explore the potential dependency among them. This is a significant advantage over an application-driven context processing approach, in which each application handles the requests of its own separately. Shared evaluation can be exploited in many different levels. For example, several applications might be interested in the same contexts. Even for different contexts, intermediate results within a processing pipeline can be partially shared.

A platform can also take advantage of potential dependencies between repeated monitoring processes. The monitoring process for each data input can be considered not as an independent one but as a part of successive ones for continuous data streams. Thus, the processing steps can be streamlined to utilize such dependencies and accelerate the continuous processes. A useful approach in this regard is to exploit the locality in sensor data streams. Consecutive sensor readings usually show gradual changes, especially for sensing of physical phenomenon. Much computation

would potentially be saved in successive processing steps with a crafted design exploiting the locality.

MobiCon develops a shared index of context monitoring requests, called CMQ-Index, and an efficient shared incremental processing method based on the CMQ-Index. Once the shared index is built for all registered CMQs, upon each sensing data arrival, all the CMQs are processed with a single evaluation. The shared processing greatly improves the performance of the context monitoring upon successive sensing data. It also adopts an incremental evaluation method and further accelerates the repeated monitoring processes. The incremental processing exploits the locality and consequent overlaps between successive evaluations. To support the incremental processing, the CMQ-Index is developed as a stateful index and remembers the state of the recent evaluation. The evaluation for a new data input proceeds quickly from the stored state.

Here we briefly describe the CMQ-Index and its evaluation. As shown in Figure 9, a CMQ-Index is created for five CMQs by building a linked list data structure. Each CMQ is already translated into feature level and thus has a range, e.g., energy > 52 in Figure 9. For the simplicity of explanation, we assume that each translated CMQ contains a single feature type. Instead of evaluating each CMQ separately, such an index enables shared evaluation upon a new arrival of feature data value. Moreover, the pointer in the table enables incremental processing by remembering the state of the previous evaluation. In the figure, the pointer tells that computation ended at N_4 at the last evaluation with data v_{t-1}. With new data v_t, the computation starting from N_4 and proceeds to N_2. While traversing from N_4 to N_2, state-changed CMQs are quickly identified, i.e., CMQ_2, CMQ_3, and CMQ_4 become false, and CMQ_5 becomes true. Due to the locality of consecutive feature values, such a traversal is often quickly done, thereby significantly accelerating repeated CMQ-Index evaluation. The CMQ-Index approach outperforms state-of-the-art query indexing mechanisms [14] by orders of magnitude [15][16]. Also, the structure is memory-efficient, since it only stores the differences between queries over successive ranges without replication.

7. ENERGY-EFFICIENT SENSOR USE

It is challenging for energy-scarce sensors to conduct continuous operations such as sensing, filtering, feature extraction, and transmission to perform context monitoring. Such operations can quickly drain the energy of battery-powered sensors and shorten the lifetime of the sensors, thereby significantly compromising the full functioning of context monitoring. Thus, the mobile context monitoring platform should provide a solution to achieve energy-efficiency in the use of sensors.

There has been much research effort to achieve energy efficiency in sensor and mobile systems. Approaches employed by the previous research include hierarchical sensor management, energy-fidelity tradeoff, interest-based sensor management, sampling rate adjustment, and sensor data compression. Here we briefly review the approach and introduce the proposed solution in MobiCon.

The hierarchical sensor management reduces energy consumption by exploiting a low power tier that consumes less energy to avoid the continuous operation of a high power tier that drains much energy. Different types of hierarchical relationship have been utilized for the approach. For example, Turducken [33] exploited the hierarchy between low power devices and high power ones while Mercury [28] exploited the one between different sensors in use in a sensor node, i.e., accelerometer vs. gyroscope that consumes much more energy than accelerometer.

Techniques based on the energy-fidelity tradeoff prepare a few levels of fidelity associated with different energy consumption and alternatively use an appropriate one to save energy. For example, Eon [32] allows applications to prepare multiple program flows associated with different fidelity for different energy states, e.g., high- .vs low-power states. Similarly, applications in Levels [36] provide alternative code blocks for different energy levels. Utilizing the alternatives, runtime systems determine an appropriate level of fidelity which can be supported under the given energy availability.

The interest-based sensor management is to only activate sensors necessary to generate contexts or data only when applications are interested, and avoid unnecessary energy consumption of other sensors (e.g., Lance [34], EEMS [35]). Sampling rate adjustment is to reduce data sampling rate as much as possible without affecting recognition accuracy, thereby saving energy for sensing data [37]. In addition, sensor data compression reduces the amount of transmitted data at the cost of computation and saves energy consumed for radio transmission [38].

The MobiCon's solution to energy efficiency falls in the category of interest-based sensor management. MobiCon efficiently supports applications without activating and utilizing all the sensors by exploiting the characteristics of personal context and application's requirement for context monitoring. It employs a novel sensor control method to utilize the structure of context monitoring queries and the dependency among them. The key idea is that only a small number of sensors often suffice to make the responses for all the queries. We call a set of such sensors the *Essential Sensor Set* (ESS). The ESS changes depending on the current context and registered monitoring requests. However, the context of an individual often remains the same once it is set to a situation. Likewise, the ESS does not abruptly change. Once we know the ESS, sensors not in the ESS do not have to transmit data. The sensors can considerably reduce wireless data transmission, thereby saving much energy.

- ESS-based Sensor Control

Calculating the ESS is a complicated problem. The ESS should include as few sensors as possible to save energy. The selection of sensors for ESS should also consider data transmission rates of the sensors.

Consider a CMQ represented in conjunctive normal form (CNF) of context elements. As described before, the core of CMQ evaluation is to detect whether the states of CMQs change or not. There are three cases to consider:

- A true-state CMQ: The state of the CMQ changes to false if the state of a single context element changes to false. Thus, all the context elements should be monitored to see if the CMQ state changes. All sensors related to the context elements should be included in the ESS.
- An undecided-state CMQ: This case should be handled similarly. The states of all context elements must be checked and all related sensors should be included in the ESS.
- A false-state CMQ: Monitoring only a single context element in a false state is sufficient as long as its state remains the same. Only when its state changes do the states of the other elements need to be monitored.

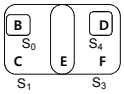
The challenge in ESS calculation is in computing the minimum-cost ESS for the false-state CMQs. However, it is the false-state CMQs that also give chances to achieve energy efficiency; as above, selecting a single context element in a false state suffices. In addition, it is also important to choose a false-state context element that is associated with the most energy-efficient sensors. The problem is formulated as the *minimum cost false-query covering sensor selection* (MCFSS) problem [18]. The problem is NP-complete, to which a well-known Minimum Cost Set Cover (MCSC) can be reduced. Accordingly, MobiCon employs a heuristic algorithm, i.e., Greedy-MCFSS (see [18]). Figure 10 shows an example of MCFSS problem and corresponding ESS result. Query A is true. Thus, sensor S_0 for the feature F_0 related to the context element of A should be in the ESS and update data. All other queries are false; for example, query B is false and its state can be determined either by S_0 or S_1. In conclusion, sensor S_0, S_1 and S_3 suffice to evaluate all the registered CMQs.

Sensor ID	S_0	S_1	S_2	S_3	S_4
Data rate	1	1	1	1	1
Feature ID	F_0	F_1	F_2	F_3	F_4
Value	7	31	2	15	72

(a) Sensor set

Query ID	Condition	State
A	$(2<F_0<15)$	T
B	$\mathbf{(F_0<5)} \wedge \mathbf{(F_1<15)} \wedge (F_3<20)$	F
C	$\mathbf{(12<F_1<18)} \wedge (1<F_2<20) \wedge (10<F_3<34)$	F
D	$(20<F_1<40) \wedge \mathbf{(7<F_3<13)} \wedge \mathbf{(48<F_4<65)}$	F
E	$\mathbf{(15<F_1<20)} \wedge \mathbf{(5<F_3<10)} \wedge (55<F_4<80)$	F
F	$\mathbf{(18<F_3<23)} \wedge (70<F_4<95)$	F

Bold character: false **(b) Query set**



(c) False-state query set
F-QSet = {B,C,D,E,F}

(d) True-state query set
T-QSet = {A}

ESS = $\{S_0, S_1, S_3\}$
(e) ESS

Figure 10. An example of ESS problem

8. APPLICATION DEVELOPMENT

● Swan Boat

Swan Boat [21][22] is a step toward applying the MobiCon framework to pervasive games. Pervasive games utilize users' various contexts and reflect their physical actions from their everyday activities. Swan Boat is designed to make treadmill running less boring. Figure 11 (a) shows a screenshot seen by the players and a team playing the game. Hand gestures are also used as input for additional game activities. For example, the players can attack opponents by punching at the same time (Figure 11 (b)). With MobiCon, developing pervasive games is much simpler; game developers only need to define the game rules and design user interfaces. In Swan Boat's case, complexities such as processing acceleration data and recognizing the motion are handled by MobiCon through a simple CMQ registration.

(a) Player screenshot **(b) Game play**

Figure 11. Swan Boat

● U-theater

U-theater is a group-interactive gaming application for public places with a large screen, like conventional movie theaters.

We suppose typical games in U-theater are likely played by tens of players at the same time as shown in Figure 12 (b). Each player wears sensors, watches the screen, and plays diverse games with body motions. By leveraging MobiCon and its APIs, developers implement the application without much knowledge of sensor data acquisition, feature extraction, and motion processing; consequently, they are able to concentrate on designing game contents. U-theater would introduce a unique gaming experience of group collaboration or competition involving physical activities. We have developed three games for U-theater: "Cheer together!", "Smash the beehive!", and "Jump, Jump!". Figure 12 (a) shows a screenshot of the "Smash the beehive!". A beehive with a lot of bees swarming around is shown on the screen. Then, suddenly there comes a moment when no bees fly. The players should take that chance to punch the beehive, and the quickest wins the game.

(a) Smash the Beehive! (b) Game play

Figure 12. U-theater

- **SympaThings**

SympaThings, an application inspired by affective computing, is a demonstration of MobiCon's wide applicability. SympaThings runs on a wearable device and controls nearby smart objects to sympathize with a person's affective context. For example, as shown in Figure 13 (b) and (c), a picture frame changes the picture inside and a lighting fixture adjusts its color (e.g., red color for the high degree of strain or yellow color for the low degree of strain). Efficient processing is crucial in the operating environment of SympaThings: high-rate data from BVP and GSR sensors, and many queries for nearby smart objects. MobiCon's shared and incremental processing is essential to satisfy these requirements. SympaThings is a collaborative project with HCI Lab of ICU and Semiconductor System Lab of KAIST.

(a) (b) (c)

(a) Use case

(b) Changing pictures in frames

(c) Changing lamp colors

Figure 13. SympaThings

9. EXPERIMETS

For experiments, we produced a data workload by collecting raw data generated by eight sensors during the daily activities of a student in campus. The sensors include five USS-2400 sensor nodes (a light sensor, a temperature/humidity sensor, and three 2-axial acceleration sensors), a GPS sensor, and two software sensors for time and indoor location. The total data rate was 291.74 Hz. In addition, we synthetically generated CMQs to simulate various monitoring conditions on different types of contexts. The number of context elements per CMQ is four and uniform distributions are applied for selecting context types and values in context elements. For all experiments, we ran MobiCon on the UX27LN UMPC. We scaled down the CPU frequency to 200MHz to validate our system under a resource-limited mobile environment.

- **Evaluation of CMQ-Index-based Context Monitoring**

We investigate the performance benefit of CMQ-Index-based context monitoring method. We compare the performance with an alternative approach, *context recognition-based monitoring method*, which models existing context-aware systems [9][10][11]. It continuously receives sensor data, processes the data to recognize contexts, and evaluates queries to detect the specified context changes. Individual queries are evaluated separately.

Figure 14 shows the average processing time per data tuple with the increasing number of registered CMQs. MobiCon shows significant processing efficiency compared to the alternative. It also scales well with the increasing number of queries. The processing time of MobiCon for 1024 queries is orders of magnitude smaller than that of the alternative.

- **Evaluation of ESS-based Sensor Control**

We examine the performance of ESS-based sensor control to demonstrate the energy efficiency. As a metric, we use Transmission Reduction Ratio (TRR), which is defined as a ratio of the reduced number of transmissions to the total expected number of transmissions. We measure TRR for 8 sensing sources deployed on five USS-2400 sensor nodes.

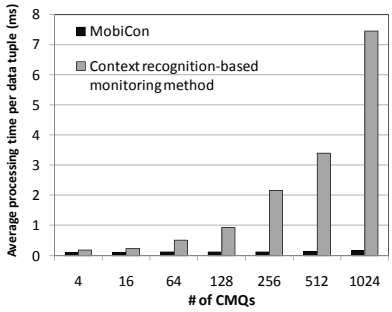
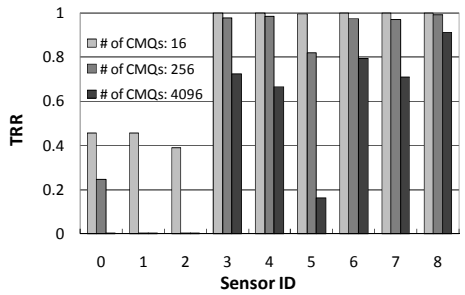


Figure 14. Average processing time per data tuple

Figure 15 shows TRR of 10 measurements for each sensing source. Acceleration sensors show much higher TRR than other sensors. Due to their high rates, ESS-based sensor control mechanism frequently excludes them from the ESS. In average, MobiCon reduces more than 90% of transmissions when the number of CMQs is fewer than 256. Also, about 63% of transmissions are reduced even with 4096 queries. As the number of CMQs increases, TRR decreases. This is because the number of true-state CMQs increases.



(0: Light), (1: Temperature), (2: Humidity), (3~8: Acceleration)

Figure 15. TRR of each sensor

10. CONCLUSION AND FUTURE WORK

New technologies and increased monitoring capacities in a sensor-rich mobile environment create a new mode of mobile context monitoring platform. The theoretical foundation and the implemented technology presented in this paper support a number of heterogeneous applications to simultaneously run and share highly scarce and dynamic resources. We have shown in this paper the disadvantage of the conventional context monitoring process is not sufficient for dynamic situations. A system titled "MobiCon" introduced in this paper will provide a new technology avenue for further exploration.

We intend to extend this work by addressing the effects of sensor noise, failure and drift on the process over time. Also, we would explore how to maintain a reliable ESS as readings become less reliable. This also would include the quality of data transmission.

11. ACKNOWLEDGMENTS

This work was supported in part by the Korea Research Foundation Grant funded by the Korean Government (KRF-2008-220-D00113), by Future-based Technology Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (20100020729), and by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2010-(C1090-1011-0004)), NSF (grant # CNS-0963793), Department of Defense- ONR (grant # N0014-08-1-0856) and Louisiana Board of Regents (grant # LEQSF-2007-12-ENH-P-KSFI-PRS-03). We are grateful to the members of the KAIST's Ubiquitous Fashionable Computer project for their collaboration on various issues. We give our special thanks to Professor Kyuho Park, Geehyuk Lee, Hoi-Jun Yoo and Ph.D. candidate at LSU: Srivathsan Srinivasagopalan. Comments from Prof. Mani Chandy, CalTech during his visit to LSU and Dr. Xin Li of LSU is also appreciated.

We also thank the reviewers and Dr. Moshi Vardi, the editor, for the useful comments and suggestions for improvement.

12. REFERENCES

[1] K. V. Laerhoven, A. Schmidt and H. Gellersen, "Multi-Sensor Context Aware Clothing," In Proc. of ISWC, 2002.

[2] O. Amft, et al., "Analysis of Chewing Sounds for Dietary Monitoring," In Proc. of UbiComp, 2005.

[3] M. Sung, C. Marci, and A. Pentland, "Wearable feedback systems for rehabilitation," Journal of Neuro Engineering and Rehabilitation, vol.2, no. 1, 2005.

[4] T. Sohn, et al., "Place-Its: A Study of Location-Based Reminders on Mobile Phones," In Proc. of UbiComp, 2005.

[5] L. Bao and S.S. Intille, "Activity recognition from user-annotated acceleration data," In Proc. of Pervasive, 2004.

[6] P. Fahy and S. Clarke, "CASS – a middleware for mobile context-aware applications," In Proc. of Workshop on Context Awareness, *MobiSys* 2004.

[7] T. Gu, et al., "A Middleware for Building Context-Aware Mobile Services," In Proc. of IEEE VTC, 2004.

[8] D. Salber, A. K. Dey, and G. D. Abowd, "The Context Toolkit: Aiding the Development of Context-Enabled Applications," In Proc. of the ACM CHI, 1999.

[9] A. Ranganathan and R. H. Campbell, "A Middleware for Context-Aware Agents in Ubiquitous Computing Environments," In Proc. of Middleware, 2003.

[10] P. Korpipää, et al., "Managing Context Information in Mobile Devices," IEEE Pervasive Computing, 2003.

[11] O. Riva, "Contory: A Middleware for the Provisioning of Context Information on Smart Phones," In Proc. of Middleware, 2006.

[12] I.H. Witten and E. Frank, "Data Mining: Practical Machine Learning Tools and Techniques," Morgan Kaufmann, 2005.

[13] G. Anastasi, et al., "Performance Measurements of Motes Sensor Networks," In Proc. of MSWiM, 2004.

[14] K. L. Wu and P. S. Yu, "Interval Query Indexing for Efficient Stream Processing," In Proc. of CIKM, 2004.

[15] J. Lee, et al., "BMQ-Index: Shared and Incremental Processing of Border Monitoring Queries over Data Streams," In Proc. of MDM, 2006.

[16] J. Lee, et al., "BMQ-Processor: A High-Performance Border Crossing Event Detection Framework for Large-scale Monitoring Applications," IEEE TKDE, vol. 21, no. 2, pp. 234-252, Feb. 2009.

[17] P. J. Lang, et al., "International affective picture system (IAPS): Instruction manual and affective ratings," Tech. Rep. No. A-4, The Center for Research in Psychophysiology in University of Florida, 1999.

[18] S. Kang, et al., "SeeMon: Scalable and Energy-efficient Context Monitoring Framework for Sensor-rich Mobile Environments," In Proc. of MobiSys, 2008.

[19] S. Kang, et al., "A Scalable and Energy-efficient Context Monitoring Framework for Mobile Personal Sensor Networks," IEEE TMC, vol. 9, no. 5, May 2010.

[20] S. Kang, et al., "Orchestrator: An Active Resource Orchestration Framework for Mobile Context Monitoring in Sensor-rich Mobile Environments," In Proc. of PerCom, 2010.

[21] Miru Ahn, et al., "Swan Boat: Pervasive Social Game to Enhance Treadmill Running," In Proc. of ACM Multimedia Technical Demonstrations, 2009.

[22] Miru Ahn, et al., "Running or Gaming", in *Proc. of* ACE, 2009.

[23] M.T. Jones, T.L. Martin, B. Sawyer, "An Architecture for Electronic Textiles," In Proc. of BodyNets, 2008.

[24] H. Harms, O. Amft, D. Roggen, and G. Tröster, "SMASH: A Distributed Sensing and Processing Garment for the Classification of Upper Body Postures," In Proc. of BodyNets, 2008.

[25] Q. Li, et al., "Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information," In Proc. of BSN, 2009.

[26] K. Chakrabarty and S.S. Iyengar, "Scalable Infrastructure for Distributed Sensor Networks" Springer-Verlag Publication Co, 2005, pp. 196.

[27] S.S. Iyengar and R.R. Brooks (Editors), "Distributed Sensor Networks" CRC Press Publication Co, 2001, pp. 1062.

[28] K. Lorincz, et al., "Mercury: A Wearable Sensor Network Platform for High-Fidelity Motion Analysis," In Proc. of SenSys, 2009.

[29] J. Lester, et al., "Validated Caloric Expenditure Estimation using a Single Body-Worn Sensor," In Proc. of UbiComp, 2009.

[30] Marc Bächlin, et al., "SwimMaster: A Wearable Assistant for Swimmer," In Proc. of UbiComp, 2009.

[31] G. Raffa, et al., "Don't Slow Me Down: Bringing energy efficiency to continuous gesture recongnition," In Proc. of ISWC, 2010.

[32] J. Sorber, et al., "Eon: A Language and Runtime System for Perpetual Systems," In Proc. of SenSys, 2007.

[33] J. Sorber, et al., "Turducken: Hierarchical Power Management for Mobile Devices," In Proc. of MobiSys, 2005.

[34] G. Werner-Allen, S. Dawson-Haggerty, and M. Welsh, "Lance: Optimizing High-Resolution Signal Collection in Wireless Sensor Networks. In Proc. of SenSys, 2008.

[35] Y. Wang, et al., "A framework of energy efficient mobile sensing for automatic user state recognition," In Proc. of MobiSys, 2009.

[36] A. Lachenmann, P. J. Marrón, D. Minder, and K. Rothermel, "Meeting Lifetime Goals with Energy Levels," In Proc. of SenSys, 2007.

[37] N.B. Bharatula, et al., "Empirical Study of Design Choices in Multi-sensor Context Recognition Systems," In Proc. of IFAWC, 2005.

[38] C.M. Sadler, et al., "Data Compression Algorithms for Energy-Constrained Devices in Delay Tolerant Networks," In Proc. of SenSys, 2006.

[39] K. Lorincz, B. Chen, J. Waterman, G. W. Allen, and M. Welsh, "Resource Aware Programming in the Pixie OS," In Proc. of SenSys, 2008.

[40] X. Liu, P. Shenoy, and M.D. Corner, "Chameleon: Application-Level Power Management," IEEE Trans. on Mobile Computing, vol. 7, no. 8, Aug. 2008.

[41] B. D. Noble, et al., "Agile Application-Aware Adaptation for Mobility," In Proc. of SOSP, 1997.

[42] G. Vert, S.S. Iyengar, and V. Phoha, "Introduction to Contextual Processing: Theory and Applications," CRC Press, Fall 2010.

[43] Kiss FFT. http://kissfft.sourceforge.net/

[44] H. Zeng, X. Fan, C. S. Ellis, A. Lebeck, and A. Vahdat, "ECOSystem: Managing Energy as a First Class Operating System Resource," In Proc. of ASPLOS, 2002.

[45] Y. Yu and V.K. Prasanna, "Energy-Balanced Task Allocation for Collaborative Processing in Wireless Sensor Networks," Mobile Networks and Applications, vol. 10, issue 1-2, Feb. 2005.

[46] R.W. Picard et al, "Toward Machine Emotional Intelligence: Analysis of Affective Physiological State", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 23, no.10, oct 2001.

[47] J. Lin, et al., "uWave: Accelerometer-based Personalized Gesture Recognition and Its Applications", In Proc. of PerCom, 2009.

[48] U. Maurer, et al., "Activity Recognition and Monitoring Using Multiple Sensors on Different Body Positions", In Proc. of BSN, 2006.

[49] H. Lu, et al., "SoundSense: Scalable Sound Sensing for People-Centric Applications on Mobile Phones", In Proc. of MobiSys, 2009.

[50] H. Lee and J. Kim, "An HMM-Based Threshold Model Approach for Gesture Recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence, 1999.

[51] G. Guo, et al., "Support vector machines for face recognition", Image and Vision Computing, vol.19, issue 9-10, 2001.

[52] S. Sitharama Iyengar, Nandan Parameshwaran, Vir V. Phoha, N. Balakrishnan, Chuka D. Okoye, Fundamentals of Sensor Network Programming: Applications and Technology. ISBN: 978-0-470-87614-5. Wiley-IEEE Press, Dec 2010.

[53] S. Sitharama Iyengar, *Louisiana State University, Baton Rouge, USA*; Richard R. Brooks; Clemson University, Distributed Sensor Networks, Series: Chapman & Hall/CRC Computer & Information Science Series, Dec 2004.

[54] Narayanan, D., Satyanarayanan, M. "Predictive Resource Management for Wearable Computing" Proceedings of the 1st International Conference on Mobile Systems, Applications and Services, San Francisco, May 2003

[55] Pradip Hari, John McCabe, Jonathan Banafato, Marcus Henry, Kevin Ko, Emmanouil Koukoumidis, Ulrich Kremer, Margaret Martonosi, and Li-Shiuan Peh, "Adaptive Spatiotemporal Node Selection in Dynamic Networks," International Conference on Parallel Architectures and

Revised paper – Dec 1, 2010.

Compilation Techniques (PACT'10), Vienna, Austria,
September 2010.