Modular and Personalized Smart Health Application Design in a Smart City Environment

Jaganathan Venkatesh, Baris Aksanli *Member, IEEE*, Christine S. Chan, Alper Sinan Akyurek *Student Member, IEEE*, and Tajana Simunic Rosing *Fellow, IEEE*

***Abstract*—The Internet of Things (IoT) envisions to create a smart, connected city that is composed of ubiquitous environmental and user sensing along with distributed, low-capacity computing. This provides ample information regarding the citizens in various smart environments. We can leverage this people-centric information, provided by the smart city infrastructure, to improve "smart health" applications: user data from connected wearable devices can be accompanied with ubiquitous environmental sensing and versatile actuation. The state-of-the-art in smart health applications is black-box, end-to-end implementations which are neither intended for use with heterogeneous data nor adaptable to a changing set of sensing and actuation. In this work, we apply our modular approach for IoT applications – the context engine – to smart health problems, enabling the ability to grow with available data, use general-purpose machine learning, and reduce compute redundancy and complexity. For smart health, this improves response times for critical situations, more efficient identification of health-related conditions and subsequent actuation in a smart city environment. We demonstrate the potential with three sets of interconnected context-aware applications, extracting health-related people-centric context such as user presence, user activity, air quality, and location from IoT sensors.**

***Index Terms*—Internet of Things, Smart health, Context-aware computation, Personalized health-care**

I. Introduction

Sensor networks and ubiquitous sensing are evolving into the Internet of Things (IoT) – sensing and actuation backed by the existing and growing Internet infrastructure [1]. This phenomenon enables more frequent and contextually-rich interactions with people and the surrounding environment. Furthermore, these environments have become truly augmented with rapidly growing available sensing and actuation devices [2], evolving into smart spaces and eventually smart cities. IoT applications in smart cities operate amid changing inputs and available compute nodes as sensors and devices enter and exit an application's domain with the goal of leveraging any available data to drive automated actuation.

The available IoT infrastructure in a smart city environment presents an excellent opportunity to improve and extend smart and connected health applications that focus on individuals. By their very nature, smart health applications exemplify the prototypical *context-aware* IoT application: they monitor users [3][4] and/or the environment [5] for important conditions and provide some actionable or visualizable output. As the more complex scenarios often involve higher-order processing[6], specialized processing or machine learning are also employed [7][8]. However, with ubiquitous sensing and actuation in the IoT, we can extend these applications to leverage data about the physical and virtual spaces through which the user moves.

To date, a majority of smart health applications focus on monitoring a patient within a fixed location (e.g. a hospital room or home) or monitoring a patient's state (e.g. with medical-grade wearable devices) [8]. The advent of the IoT allows smart health applications to focus on the intersection of an augmented environment (smart space) and the users that move through it [6]. The environment-provided context can be leveraged to inform or guide user behavior[3], while the user's presence, activities, and goals – the user's context – is used to update and further augment the physical and virtual spaces they move through [9]. While the value of the contextual data dynamically changes according to the user or environment, its type, range, and sources are always defined within an ontology.

In the absence of the IoT, the current state of the art in smart health applications is end-to-end systems tightly coupled to the initial infrastructure and platforms. This is due to a specificity of body-worn sensing systems and their associated uses [10]. However, commercial wearables that monitor similar data are prevalent as IoT devices. Furthermore, in the context of the IoT, the currently tightly-coupled applications do not promote adaptation to the changing amount and sources of data or available compute nodes. While other aspects of smart health applications are moving towards a more generic approach: unified data storage and frontend infrastructure based on electronic health records (EHRs) [11] and standardization of medical device approval [12], applications themselves remain black-box, end-to-end (monolithic) implementations that produce application-specific output.

We have previously proven that smaller, simpler functional units provide intermediate steps towards an overall application that can alleviate application redundancy and facilitate the use of general-purpose machine learning [13]. Smart health applications fall within the scope of the same problem. In this work, we present connected health applications comprised of general purpose functional units (context engines). This modular approach can provide similar complexity improvements due to the processing complexity, and can reduce the impact on accuracy. We build three case study applications,

Jaganathan Venkatesh was with the Computer Science and Engineering Department, UC San Diego, CA, 92093 USA e-mail: jvenkate@eng.ucsd.edu.

Baris Aksanli is with the Electrical and Computer Engineering Department, San Diego State University, CA 92182 USA e-mail: baksanli@sdsu.edu

Christine S. Chan is with the Electrical and Computer Engineering Department, UC San Diego, CA, 92093 USA e-mail: csc019 @eng.ucsd.edu.

Alper Sinan Akyurek and Tajana Simunic Rosing are with the Department of Computer Science and Engineering, University of California San Diego, CA, 92093 USA e-mail: aakyurek,tajana@eng.ucsd.edu.

combining medical-related sensing (user activity and local air quality) with other IoT sensors in the space around them (potential to increase physical activity and alternate routing for respiratory health) and smart meter data to infer user presence. They all take advantage of the ubiquitous sensing and many distributed compute nodes available in the IoT space. They also demonstrate how our approach can help "grow" additional applications using existing infrastructure.

This motivates a new approach to smart health applications in the context of the IoT: share common processing to transform raw user data into intermediate context. Furthermore, we can expand these applications to take advantage of secondary sensors in the environment for correlated processing, and the more numerous IoT computational units (e.g. smartphones, health monitoring devices available throughout a smart city environment) associated with each user to perform distributed, user-specific output generation. The overall advantages of this approach are: improved latency, reduced overhead, and general-purpose machine learning. For smart health, this translates to faster response to critical situations (e.g. more timely notification of air quality and alternate routing); more efficient processing of health monitoring (e.g. implementation on the low-capability wearables and mobile devices); and ease of creation and expansion of health applications (e.g. growing the scope of health monitoring using the existing infrastructure).

II. Related Work

Pervasive sensors gather raw data from a diverse combination of data sources, including sensors and user-supplied or high-level context processed from mobile and computing devices. Analog sensor data has to be at least digitized and preprocessed before software can use it as meaningful input. In the Internet of Things, most data goes through several levels of abstraction, combination, or distillation to produce a description of the environment (and its users) with discrete, semantic states. This higher level context is used for visualization (e.g. quantified self [14], vehicular safety [15]) and actuation (smart spaces [16], ubiquitous computing [15], medicine [7], e-learning [17] and user behavior tracking [18]). In exchange for raw data precision, discretized context facilitates intuitive reasoning and reuse across applications. These current context-aware applications are individual deployments that rarely share infrastructure, code, or data natively. Practically, this end-to-end development approach results in a disorganized data space, necessitating the use of ontologies to maintain a unified, regulated data representation.

Pervasive sensing and computing in the IoT is facilitated by learning and reasoning within applications to appropriately transform input data into output context and actuation. For example, when streaming data from human subjects, sliding windows of the continuous data must be smoothed and preprocessed before inputting into an analytic or modeling framework. K-means clustering is a prevalent way to automatically relate low-level data into high-level context [14]. Reinforcement learning (RL) invites users who are already involved in sensing and actuation to reinforce and guide the system towards better accuracy and intuitive actuation. Madhu et al. [17] use constraint reasoning to describe a daily plan and RL to find optimal customized reminders for a cognitively or orthotically impaired user. Rashidi et al. [3] perform unsupervised learning over low-level sensor event sequences to extract patterns that represent high-level activities. They focus on a specific implementation for the smart home over a known set of activities, but we propose a framework and algorithms that can perform a similar level of data translation and actuation in a domain-independent manner.

The goal of the IoT application layer is that it provides an interface between sensing and actuation in the IoT. A key takeaway is that applications operate in a dynamic space: mobile sensing devices (e.g. wearables) and compute units (e.g. mobile phones) enter and leave the domain of a particular application [19]. Some leverage ontologies to provide platform independent organization of applications: black boxes that transform input data into output data for a specific application [20]. Perera et al. [1] reinforce this view, providing a comprehensive overview of context-aware applications covering fifty publications over the last decade. They view applications as multi-input, multi-output (MIMO) units composing similar data transformations to obtain output context information.

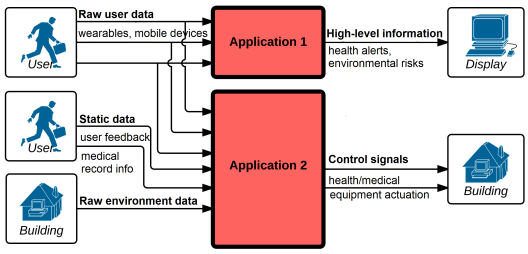
We identify three major challenges with the current view: **1)** There is significant processing redundancy: different applications using the same input may repeat the same computation. For example, user occupancy data serves both home security and grid automation applications, and may be independently computed by both. *We instead expose the output of this computation for reuse.* **2)** The complexity of IoT applications grows rapidly with input and output spaces. This in turn increases the computational cost of machine learning (ML) algorithms, whose complexity is dominated by the number of inputs. This in turn forces application-specific implementations that cannot be reused. *By reducing the number of inputs per functional unit and enforcing a single-output approach, our approach facilitates the use of ML.* **3)** Without effective reuse of data and functionality, the scalability of IoT applications is severely limited. Large application functional units (see Figure 1 (a)) preclude a general approach to distributed computation, modularity, and reduction of complexity. *Our approach focuses on modularity, which in turn creates applications that can be readily distributed or parallelized.*

III. Context-Engine Design

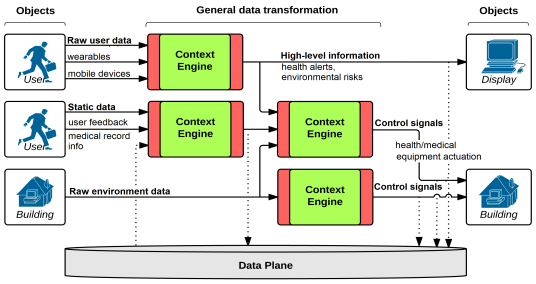
We design smart health applications as a hierarchy of common multiple-input-single-output (MISO) functional units called context engines to improve reasoning and scalability while reducing the data redundancy across applications, and accomplishing the same functionality as the previous monolithic multi-input multi-output (MIMO) units. In exposing intermediate data and making applications share them (Figure 1b), we reduce the complexity and improve the scalability of other applications in the larger infrastructure. The improvement in scalability may come at the cost of accuracy, but we quantify the error and show that it can be minimized by the simple expedient of intuitive design in our initial work [13].

IoT applications consume data about both physical and virtual system entities. This data, from heterogeneous sources

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2712558, IEEE Internet of Things Journal



(a) The state-of-the-art: monolithic MIMO implementation



(b) Our implementation: Applications publish intermediate context for reuse. MISO functional units perform general statistical learning

Fig. 1: Red represents developer effort; green represents generalized data transformation provided by our system

including sensors, social media, and even manually submitted by users is raw and noisy requires processing by applications to be filtered and distilled into usable information. Additionally, from the input data, applications need to extract context: high-level abstracted data. In the IoT, context tends to be human-centric classifications (e.g. location, activity) that are important to many different applications [21]. Black-box implementations of applications from raw data to output mask both types of processing output (preprocessing and common intermediate context) from other applications, which leads to redundancy in computation. Our proposal of a hierarchy of functional units in place of monolithic implementations trades off compactness for versatility. A hierarchical approach breaks down a single application into multiple functional units, increasing organizational complexity, as shown in Figure 1b compared to Figure 1a. Although serializing the process can increase latency if a highly compact algorithm was expanded, it can also expose intermediate output for reuse by other applications, thus reducing compute redundancy in the system. We previously proved that such an approach decreases overall compute complexity and improves system scalability for data processing that exceeds 2^{nd}-order complexity (please see our previous work for more details [13]). As we show in our case studies later in this paper, even the simplest applications exceed quadratic complexity when machine learning is involved. We also provide a detailed analysis of preserving the accuracy using our approach in our previous study [13]. Additionally, splitting single-step applications into small functional units (each with fewer inputs, simpler logic) facilitates a generalized data transformation through machine learning.

We leverage ontologies to specify the interfaces to each of the context engines [13] and, uniquely, to also drive data transformation. We define a context variable as the individual input or output data unit, and leverage the variable's space representation of ontologies to define the domain/range of the variable, and subsequently, the context engine:

- **Discrete context variables** must have a countable set of possible states. This is a requirement for the data processing algorithms to map to input or output states
- **Continuous context variables** require a range of possible values, to allow the algorithms to perform clustering over this range. Different ontologies make allowances for multi-interval and other continuous representations.

While current monolithic applications may have internal modularity and parallelism, they are hidden from the rest of the system. The MIMO implementation of a current IoT application can be explicitly broken down into several context engines, which decompose its internal functionality into smaller, MISO functional units. The composition of the context engines delivers the same outputs as the original application (Figure 1(b)). This reduces the complexity of each context engine, as each performs less processing than the single-stage engine, and improves the scalability, as each requires fewer inputs and produces fewer outputs. For IoT applications, this increases the overall versatility, as the now-visible functional units and the resulting dependency graph can be deterministically or automatically distributed and parallelized among available compute nodes by the IoT management system. Finally, the intermediate outputs of the distributed approach provide additional context that can be reused by other applications, reducing their computational complexity and consequently the compute redundancy of the system at large.

Generalized Data Transformation: Our approach, a modular multi-stage context engine, results in more functional units (FU) per application. An important consequence is that each FU that composes the application is a simpler translation of input data to a single output. This enables the use of a general data transformation in each context engine in place of application-specific code. Thus, a context-aware application can be created by specifying the inputs and output of each FU alone, and allowing the data transformation algorithm to incur the processing overhead generating and training a model based on input and output observations.

We leverage the ontologies that are already present in the current state of the art of IoT middleware. From a data standpoint, they regulate inputs and outputs of applications. Applications that participate in the system must enforce the ontology's specification: discrete variables must provide a set of possible states to populate the probabilistic condition tables; continuous variables must specify a valid range of values that can be clustered. We can exploit this ontological information for machine learning algorithms that clusters results based on the space of the input and output variables, as well as determines the training space and list of prior observations.

Matrix-based stochastic learning models express potential data dependencies as a system of equations. Some use predefined notions about the inputs to establish linear or nonlinear equations, while others start with a linear combination of the inputs with unknown coefficients. Over time, observed input and output data is gathered until the coefficients can be trained

and a model generated. Since complex relationships can exist among the input data for an IoT application, and a purely linear model may not be sufficient [22], several works [23] [24] [25] implement learning by considering higher orders and time correlation. In our current implementation, we leverage TESLA (Taylor Expanded Analog Forecasting Algorithm), a statistical learning model that can be fully generalized, as the data translation algorithm in our context engine. It provides efficient model generation: $O(n^{\alpha})$, where n is the number of inputs and α is the function order of the Taylor expansion. The generic function of this expansion is established as follows:

$$\sum_{i=o}^{n} C_i x_i \quad (1^{st} order) \tag{1}$$

$$\sum_{i=0}^{n} \sum_{j=0}^{i} C_{ij} x_i x_j \quad (2^{nd} order), \quad etc. \tag{2}$$

where C_{ij} represents coefficients learned with observations, and $x_0 = 1$ (the constant factor). The resulting equation is $Ax = B$, where A is the matrix of input observations; x is the vector of coefficients, and B is the vector of output observations, each entry correlating with the corresponding row of A, and solved by least squares estimation. Higher function orders are able to represent more accurate correlations between input variables, but they require exponentially more training samples with respect to α for example, 1^{st}-order (linear) functions only require n samples, whereas 2^{nd}-order functions require n^2 samples.

We use TESLA in our study for its versatility, but other statistical learning approaches such as Bayesian Networks [26], Hidden Markov Models (HMM), and Artificial Neural Networks (ANN)[24] can be substituted as the learning algorithm in other context engines. We select algorithms that have low-enough computational complexity to run on the devices found in the smart health infrastructure: embedded sensing systems, local aggregators (health monitoring gateways), and the mobile devices patients and providers use for monitoring.

IV. Case Study 1: Occupancy Detection

Our first case study demonstrates the context engine approach and investigates an application that is extremely useful to smart cities and smart health-care domain: occupancy detection. In a smart health-care application, it is very important to know the presence of a person in a specific room/environments without their input. Our approach presents a unique opportunity: to train and learn the behavior of different people/residences using a shared infrastructure.

A. Context Engine Setup

Input Data: For input data, we use plug load information from appliances. With the advent of smart metering and techniques such as residential energy disaggregation, obtaining appliance- specific information is relatively straightforward. We use the MIT Residential Energy Disaggregation Dataset (REDD) [27] hertz-scale energy traces for each appliance.

Application: We aim to translate commonly available plug load data into room-level user occupancy. Determining this output is very useful for smart health applications (locating potential health hazards), and the distributed nature of the problem (different models for different rooms) empowers the use of multiple context engines.

Context Engines: In order to go from plug loads to user activity, we break up the problem into two steps 1) determining appliance usage and 2) determining user activity based on 1). Figure 2 illustrates both the sequential context engine configuration (a) and the single-stage application (b) to determine output context. For a set of n appliances and m rooms, there is a set of n context engines. Each first-stage context engine is a single-input, single-output $(O(1))$ engine consuming the raw power data from its corresponding appliance as input, and trained on human-observed binary output on appliance activity. The second stage transforms the output of all the first-stage context engines (active appliance usage) to a binary representation of whether or not a room is being actively used. These second-stage context engines, with n inputs, have a complexity of O(n) for generating the output context, where a is the functional order. The single-stage, consolidated application represents the current state of the art, taking in the same input to produce the same output. It will be used for comparison of scalability, complexity, and accuracy. In this case, the single-stage application has a complexity of $O(n^{\alpha})$. We use up to 172,800 appliance samples (2 days' worth) from REDD, and test against 86,400 (1 day) samples.

B. Results

The sequential context engine consists of n 2^{nd}-order context engines for the first stage, each consuming one input (appliance raw data) and producing binary appliance usage context. The second stage has m 2^{nd} context engines, one for each room, with n inputs, from each first stage engines. The result is $n * O(1)$ complexity for the first stage and $m * O(n^2)$ for the second stage, and an overall complexity of $O(m * n^2)$. The single-stage version has exactly the same configuration as the second stage alone and consequently, the same complexity, at $O(m * n^2)$. At the same complexity, our approach improves execution overhead. Each context engine iterates when a new input observation is recorded. However, the second stage sequential context engine reacts only to changes in binary input data, which is much less frequent than the corresponding changes to the raw data. Table I highlights the number of computations performed in the single-stage and sequential context engines for the fridge. The sequential application performs more individual computations because there are more context engines, but it exhibits only 31% of the latency of the single-stage context engine despite the additional throughput. This is because of the nature of the computations: the sequential context engine offloads the processing of raw data to the low-overhead $(O(1))$ first stage. The single-stage context engine has to perform over 73000 $O(n^2)$ computations. The sequential application requires only 23% of the intensive $O(n^2)$ computations as the single-stage one, and thus, can complete the work 69% faster.

Then, we investigate the accuracy changes, with increasing functional order, and more training data available. Figure 3

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2712558, IEEE Internet of Things Journal

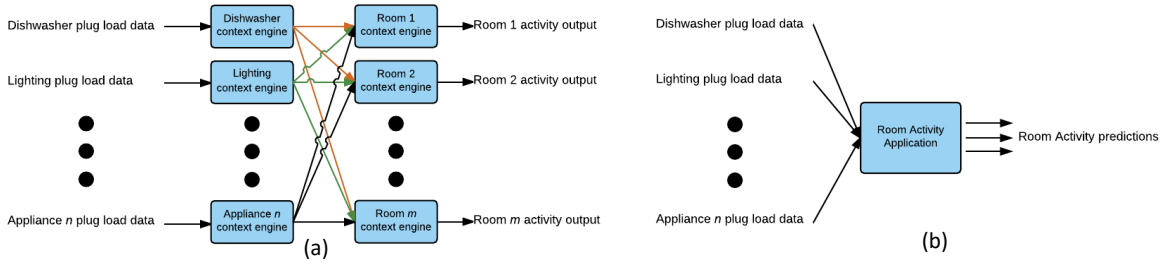


Fig. 2: (a) Sequential and (b) consolidated single-stage applications for in-room occupancy prediction

Refrigerator	Number of computations		Total latency
	1^{st} **stage** $O(1)$	2^{nd} **stage** $O(n^2)$	
Sequential	73428	17376	0.42 sec
Single-stage	-	73428	1.34 sec

TABLE I: Execution overhead based on iteration count for the context engines associated with the refrigerator

Fig. 3: Accuracy comparisons of the first stage context engine against different training set sizes and functional order

shows accuracy comparisons on the first stage of the context engine for the microwave. There is a clear correlation between increasing functional order and accuracy, since the complex relationships between the inputs and output are described better with higher functional order. Similarly, error decreases with an increasing training set, as issues with under-fitting the function to the inputs and output are mitigated with more data. Other appliances showed similarly low error counts with our chosen order (2^{nd}) and training set (2 days).

We now investigate the output prediction error in Table II. Error is calculated based on the percentage of instances where the model-derived boolean output differs from the expected output value. Table II depicts this error for kitchen-related appliances. These appliances were trained with 10,000 unique observations culled from 2 days' worth of data, and demonstrate low error near-unanimously.

Appliance	Prediction Error (%)	Appliance	Prediction Error (%)
Kitchen Outlets 2	32.8	Oven 1	1.4
Kitchen Outlets 3	1.4	Oven 2	1.4
Kitchen Outlets 4	0.6	Refrigerator	0.8
Microwave	4.0	Stove	0.0

TABLE II: Prediction error for 2^{nd}-order context engine

We now investigate the output accuracy of both the sequential and the single-stage implementations of the same application. An important consideration is that the sequential application contains more opportunities for training, as each context engine can be trained individually. For example, the kitchen activity-detection application contains 9 individual context engines (8 first-stage engines listed in Table II and 1 second-stage engine that takes in all those outputs and translates them to the kitchen activity output), each that can be trained independently. Conversely, the consolidated application has only one context engine (see 2b). If we train each context engine in both applications with the same number of inputs, the sequential will consume and train on 9x the data of the single-stage. Alternatively, we can train the sequential application such that the total number of trained observations matches the consolidated application, that is, provide each of the 9 context engines with 1/9 (10000) unique observations over 2 days compared to the single-stage (90000 unique observations, 7 days). We investigated both cases: training each context engine with the same number of observations (row 1) and training the overall application with the same number of observations (row 2). Table 3 compares the results of the two engines for the kitchen:

Application type	Prediction Error (%)
Sequential (same # obs/context engine)	2.4
Sequential (same total # observations)	34.3
Single-Stage	17.4

TABLE III: Output accuracy comparison for Kitchen Activity between the sequential and single-stage context engines

The preprocessing in the first stage and discretization of the raw appliance data results in significantly reduced (15%) error in the second stage. When relying upon the same total number of observations, the single-stage application provided better accuracy because sequential application's context engines undersampled and aggregated noise and error in the second stage. As more input and output observations are accrued, the sequential context engine improves its prediction to 2.4%, while the single-stage application, which only allows additional training on the output variable, only improves to 12.3%. The coefficients show that the additional training of intermediate context improved the output accuracy.

V. Case Study 2: User Activity

Determining a user's health can be highly dependent on his/her physical activity [6]. We investigate both applications that output user activity predictions and applications that use activity information to provide output actuation for a particular space/domain. The latter application is used to determine the potential for a location to be used for activity or exercise.

A. Context Engine Setup

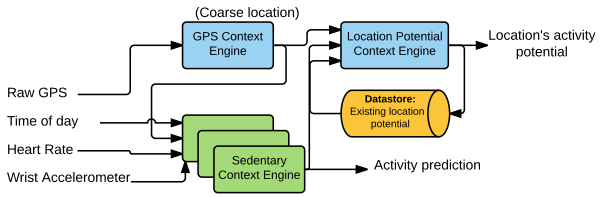
Input Data: We operate on the UCSD Personal Activity and Location Measurement System (PALMS) dataset [28]. With the prevalence of wearable and mobile fitness trackers, this dataset provides a comparable means of high-fidelity wearable data: sub-hertz GPS and heart rate data, and 30Hz data from wrist and hip accelerometers for 40 individuals over a week, with the actual activity annotated through observation of the individuals throughout the timeframe. The activities fall into four categories: the activity itself; the posture of the participant; whether it constitutes social interaction; and indoor/outdoor. The activity itself is chosen from a set, including eating, TV, leisure, sports, exercise; while posture and indoor/outdoor are additional binary values associated with but not mutually exclusive to each activity.

Applications: We considered two different connected health applications that could take advantage of both wearable personal monitoring and available spatial monitoring. The user-centric application is activity prediction, translating the raw sensor data from each particular into meaningful high-level activities for human-readable record-keeping and medical analysis. The second application is space-centric. We make use of the same wearable personal data to track available user heart rates, correlated to the location, and compute that location's potential of raising any given user's heart rate. This can be publicized to new users entering the system, as a suggestion for where they can go to improve their heart rate (the gym, as an obvious example). A potential extension is to keep track of certain activities, and make recommendations of healthy activities in a particular space.

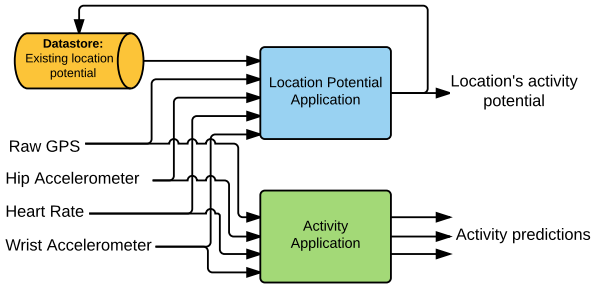
Context Engines: Current state of the art applications use all the available data as input, and produce the activity potential and activity prediction. Our approach modularizes the problem into three sets of context engines: generating coarse GPS location, detecting activities, and identifying the activity potential for each location. In keeping with the MISO principle, we allocate a separate context engine for each perceived activity. Also, we assign activity detection context engines for each of the 40 users of the application, leveraging the availability of compute units on each user's fitness trackers, capable of computing personalized activity detection.

Secondly, as GPS information is important to each application, but the raw GPS data is too fine-grained for either application, we introduce a GPS context engine, which outputs a coarser latitude/longitude reading representing a larger physical space. The reference data to train this context engine is derived by spatially clustering the raw GPS data and using the northwest point of each cluster. The output prediction is a boolean value whether or not an activity is detected. Similarly, a location's activity potential is a boolean. Both values are trained using the available annotated data, or ground truth, from the PALMS dataset.

Figure 4 illustrates the selected configurations for both the sequential context engine (a) and the state of the art single-stage application (b). In the former, the GPS context engine's output supplants the raw GPS latitude/longitude data in both applications, though each activity's context engine still



(a) Sequential context engine applications



(b) Applications for activity and location potential detection

Fig. 4: Context-engine applications

requires the speed and number of satellites found from the raw GPS data. The second stage transforms the available input from the original data sources and the GPS engine to a boolean representation of the respective activity. While Figure 4 shows a fixed number of inputs, the number can vary depending on the data available from each fitness tracker. Finally, the third stage is the primary location potential application. It only uses intermediate data produced by the other context engines (including feedback from itself). While the data from the GPS context engine and the feedback are fixed, it is possible for the activity detection context engines to grow in number as more activities are added to the system. The compute complexity is $O(kn^{\alpha})$, where k scales with the number of activities that are detectable and α is the function order.

For each activity's context engine, trained with a different input and output set, we observed the varying output accuracy using 1^{st}-, 2^{nd}- and 3^{rd}-order functions under the TESLA algorithm. Each context engine was then finalized at the lowest functional order that yielded marginal accuracy improvements.

B. Results

Both of the subsequent context engine applications use the GPS context engine in their first stage, as its output is consumed by following stages. The single engine was tested across different functional orders of the TESLA algorithm. As the data processing is a linear function, it performs well with first-order ($O(1)$ complexity), gaining only marginal improvements for second- and third-order functions, with a mean absolute error of 5%, as shown in Figure 5.

The context engines for the various activities are used by both applications, and vary in complexity. Some activities (indoor 1^{st} order; sedentary, biking, exercise, in-vehicle 2^{nd} order) are more readily predictable than others (walking, watching TV, eating 3^{rd} order). Figure 6a shows the change in the mean absolute error for different functional orders. Some activities have highly similar output given the available sensor

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2017.2712558, IEEE Internet of Things Journal

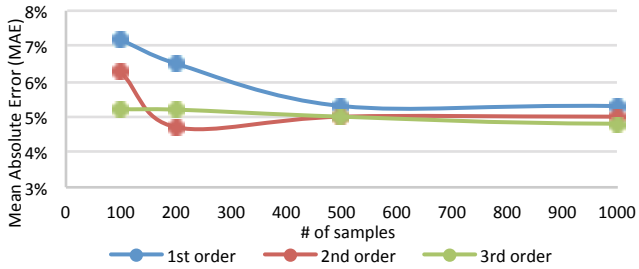
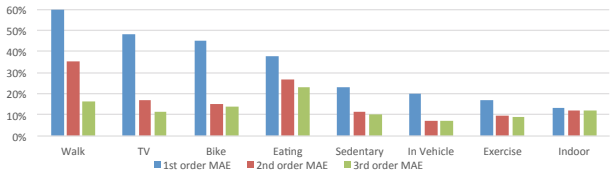
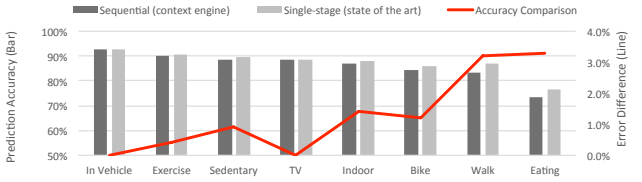


Fig. 5: Mean Absolute Error of GPS Context Engine over functional order and sample size



(a) Mean absolute error (MAE) for each activity context engine across different function orders



(b) Accuracy comparison between the context engine and the single-stage activity applications (bar) with accuracy difference (line)

Fig. 6: Results for the first case study

data, and require a more complex function to differentiate them from others. For example, eating shows marked similarities to indoors, and can report false positives even at 3^{rd} order. Furthermore, some activities are highly correlated with sensor inaccuracy: walking incurs accuracy issues due to reduced GPS reliability. Other activities can be determined with lower function order: for example, determining that the user is indoors can be determined with just a 1^{st}-order function.

The final stage of the sequential context engine approach is the location potential context engine. It incorporates the activity potential for the nearest known location found in the datastore (Figure 4). This last input dominates the results, and a second-order function, $O(n^2)$, is sufficient to generate accurate output (9.2% MAE). Contrastingly, the single-stage approach requires 3^{rd}-order TESLA algorithms $O(n^3)$ for each application to produce the same outputs.

Since the context engines in the sequential approach are composed to form applications, each later stage will only react to changes in the relatively high-level output of the previous stages. This results in the more complex 2^{nd} and 3^{rd} stage context engines performing fewer total operations than the single-stage approach. The modularized functional units offload the processing of raw GPS to the low-overhead, very fast ($O(1)$) GPS context engine. The single-stage context engine, however, has no choice but to perform $O(n^3)$ computations

In Vehicle Prediction	Number of computations		Total Latency
	1^{st} stage $O(1)$	2^{nd} stage $O(n^3)$	
Sequential	3670	1823	0.24 sec
Single-stage	-	3670	0.37 sec

TABLE IV: Execution overhead based on iteration count for the context engines associated with the In-Vehicle activity

Function Order (Context Engine Applications)	Normalized Latency
1^{st} (Indoor)	0.38
2^{nd} (Sedentary, Bike, Exercise, In Vehicle)	0.49
3^{rd} (Walk, TV, Eating)	0.65

TABLE V: Sequential application latency vs. function order

for every minor change in GPS data. Table IV highlights the number of computations performed by the single-stage and sequential context engines for the 4,000-input (one day) test set for the In-Vehicle activity. We only present a representative activity in this table for clarity, but this trend is observable for other activities as well. Ultimately, the sequential application requires fewer than 50% of the time- and compute-intensive $O(n^3)$ computations as the single-stage implementation, and can complete the work 35% faster. This difference in latency results in a speedup of up to 2.6x. For smart health applications, this translates to important improvements: reduced processing latency results in faster response time to critical situations and timelier reaction to changes in the users' context as well as that of their environments. Table V shows the latency of the context-engine-based applications normalized against their single-stage counterpart.

We now compare the accuracy of both applications between the sequential and single-stage counterparts, with the activity prediction accuracy highlighted in Figure 6b. The context-engine approach demonstrates very similar accuracy to the current state of the art, with the highest error difference being 3.3%. The location potential application is an ideal example for contrasting the context engine from the state of the art, because instead of using the raw input data, it relies only on intermediate data the outputs of the activity and GPS context engines. This is a key motivation for the modular context engine approach, because we expect to be able to grow applications from existing data rather than creating entirely new, independent ones. Despite relying solely on intermediate data, there is only a 5.7% reduction in output prediction accuracy: 79.9% for the consolidated approach vs. 74.2% for the context engines. For smart health applications, accurately identifying the critical output is important, as it could have life-threatening consequences. Our approach enables efficiency in processing and incremental extension of a connected health suite with little impact on accuracy. As the quality and number of medical sensors improve over time, an application can seamlessly integrate more accurate and numerous data to improve accuracy, rather than redesign and redeploy.

VI. Case Study 3: User-Centric Air Quality

Another important metric for smart health applications is air quality, which has connections to respiratory health and asthma [5]. We investigate an application that uses environmental information to generate user-centric health-related information.

While air quality information is useful as a static metric, it does not directly impact the users throughout their day. Citisense [5] and other citizen-driven sensing efforts have provided the ability for users to measure air quality in areas that they frequently occupy. This information is useful when consumed in a timely manner, but not when the data or location is not relevant to the user. This case study develops and deploy a user-facing air-quality application that automatically builds on the collected data from users mapped to the relative location of the current user. We compare a custom, single-stage application with our modular context engine approach.

A. Context Engine Setup

Input Data: We operate on air quality data collected from wearable Citisense sensors, which provide raw parts-per-million/billion concentrations of NO_2, CO, and O_3 as well as the The AQI data is correlated with timestamps and GPS locations obtained from the wearers' mobile phones.

Applications: We consider two related applications that share data: one specific to each user, and one specific to the spaces in which the users move. The location-centric application produces the location's AQI a value of the location's current air quality. The user-centric application produces route guidance for a user directing their route to a healthier path. This forms a consolidated IoT environment users whose sensor data impacts the spaces around them, and conversely, smart spaces providing actuation to the entering users.

Context Engines: The single-stage approach (Figure 7(right)) takes in all the available data as input, and produces the AQI and alternate route. Our approach modularizes the problem into four sets of context engines: generating coarse GPS location, a time-of-day offset, the current AQI, and the alternative coordinate. While our initial application has relatively few inputs to generate AQI, we can include other related input context variables such as air temperature and humidity [5]. In keeping with the MISO principle, our approach (Figure 7(left)) allocates a separate context engine for each. We create an alternative-coordinate context engine for each user, leveraging the assumption that the users' devices (which produce their GPS coordinates) are an embedded system capable of computing personalized route recommendations.

We reuse the same coarse GPS context engine from the previous case study, truncating the least significant digit and having each value represent a larger physical space. An independent day offset context engine uses the sensing timestamp and generates a numeric offset for each day.

The AQI context engine outputs the predicted air quality at a location at a given time. Typical citizen-driven sensing depends on the availability of nearby sensor nodes [5], but with a composition of context engines, we can provide AQI even if local sensing is not present. We provide a separate output application that only makes use of the existing infrastructure: an alternate route (to avoid poor-AQI areas) as a direction from the user's current location. Our sample application is representative of others that can further use this data for actuation, such as controlling air handlers or duty-cycling pollutant-heavy machinery to improve a user's health based on their sensitivity. Both outputs for our applications are trained using data from the Citisense dataset [5] as ground truth.

The choice and number of context engines are determined in the previous section. From this point, building the health applications is simply a matter of asking health professionals to specify the relevant inputs and outputs of interest to each context engine. Figure 7 illustrates the selected configurations for both the sequential configuration (left) and the state of the art (right). The single-stage, monolithic black-box implementations, simply require all the available input data.

The sequential approach can be designed more judiciously. The GPS context engine's output supplants the raw GPS latitude/longitude data in both applications. Each activity's context engine requires the current time, one to predict the AQI at a given time, and the other to direct the user to a different location. As the GPS context engine has only one input, its complexity for transforming input data to output context is constant $O(1)$. The day offset context engine, with two inputs, has a complexity of $O(2^\alpha)$, where α is the function order.

The second stage of the sequential activity application transforms the available input from the day offsets and the coarse location engines to a AQI value. Each of these second-stage context engines have n inputs and α computational complexity of $O(n^\alpha)$ for generating the output context.

Finally, the third stage is the entire location potential application, as it *only* uses intermediate data produced by the other context engines (including feedback from itself). While the data from the GPS context engine and the feedback are fixed, it is possible for the activity detection context engines to grow in number as more activities are added. The compute complexity is $O(kn^\alpha)$, where k scales with the number of activities that are detectable and α is the function order.

The single-stage consolidated applications are used for the complexity and accuracy comparisons formalized in previous sections. They are also run using a generalizable data transformation that is defined by a polynomial function order. Since both applications take in all the available inputs, their complexity is similar to the second-stage context engines: $O(n^\alpha)$. As each context engine uses different inputs and generates different outputs, we tested each with the TESLA algorithm up to 3^{rd}-order functions, using the lowest order after which accuracy improvements were marginal. From the Citisense dataset, we extracted correlated input data uniformly distributed over a range. To test the impact of the number of samples on functional order, we vary the number of samples up to 8,000 (two days), and test against 4,000 (one day) samples.

B. Results

As each context engine uses different inputs and generates different outputs, we can test each independently, using TESLA up to 3^{rd}-order functions, and determining the functional order after which accuracy improvements are marginal. Both sequential applications use the GPS context engine in their first stage. As each engine (one for latitude, one for longitude) only takes a single input, it has a complexity of $O(1)$. The AQI context engines are used by both applications, and are implemented by 3^{rd}-order TESLA functions for an

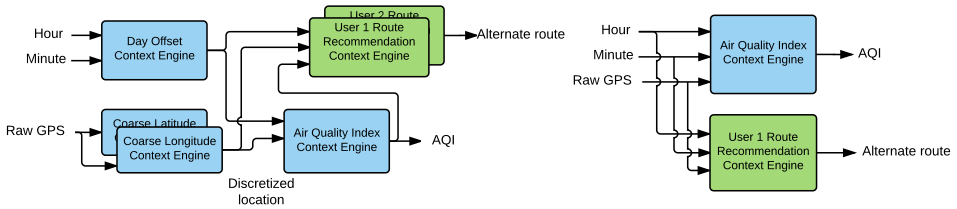


Fig. 7: The sequential (left) and single stage (right) context application for AQI and alternate routing.

	Number of computations per CE			Total latency
	1^{st} Stage GPS CE $O(1)$	2^{nd} Stage AQI CE $O(n^3)$	3^{rd} Stage Route Rec. CE $O(n^3)$	
Sequential	9793	9351	9351	1.24 sec
Single-stage	-	9793	9793	1.49 sec

TABLE VI: Execution overhead based on iteration count for AQI context engine. Note that CE = Context-engine.

overall complexity of $O(n^3)$. The final stage of the sequential context engine the route recommendation one consumes outputs generated by the GPS, time, and the AQI context engines, to generate an alternate route for the input coordinate. This is implemented as a third-order ($O(n^3)$) context engine. Each application in the single-stage approach has the same configuration: a consolidated block with $O(n^3)$ complexity.

Even with the same worst-case complexity, the sequential approach has smaller overhead: all context engines recalculate their output whenever a new input observation is recorded. However, the second stage of the sequential context engine reacts only to changes in coarse location (output from the GPS context engine), which is much less frequent than the correspondingly subtle changes to the raw GPS data. Table VI highlights the computations performed by the single-stage and sequential context engines for the 10,000-input test set.

The sequential application comprises multiple stages of context engines, which all perform their own data transformations. This naturally means that, overall, the sequential application performs more total computations than the counterpart single-stage approach. However, it exhibits only 83% of the latency of the single-stage context engine because the sequential application offloads some processing to the low-overhead, fast ($O(1)$) GPS context engine. The single-stage context engine has no choice but to perform every $O(n^3)$ computation.

The impact of these improvements is significant for health applications, where fast response time is important. From the results above, for example, the quarter-second delay can be enough to allow a walking user to enter the poor-AQI space. The context engine approach only performs the expensive operations as needed, providing better latency and more timely response (e.g. only calculating and providing alternate routes when necessary), and its relative efficiency allows it to be deployed on smaller, less intrusive embedded devices.

The mean absolute output prediction error (MAE) for the AQI context engine is 12.3% whereas the single-stage engine performs worse, with 17.2% MAE. The single-stage application, which has to deal with considerably more, finer-grained GPS locations, is less capable of handling them as accurately. We also investigate the consolidated route recommendation application compared to the equivalent sequential context engine. This is an ideal example for comparison, because instead of using the raw input data, the context engine relies on only the intermediate data the outputs of the time, GPS, and AQI context engines, a key motivation for the modular context engine approach. The single-stage application suffers the same issue with maintaining more fine-grained GPS locations, but has the benefit of all the raw data. The two applications have comparable accuracy, with 16.2% error for the single-stage, and 15.9% error for the sequential approach.

VII. Conclusion

This paper presents a versatile and flexible approach (context engine) to integrating smart health applications with the connectivity of the IoT, facilitating the use of machine learning and improving the complexity of context-aware designs. Our context engine approach to IoT applications exposes shareable intermediate context, and increases reusability while reducing computational complexity. For smart health, this allows designing more complex health applications with limited resources, using sensors both on the user and in the surrounding spaces. We implement our approach with three interconnected case studies, demonstrating the versatility of context engines (occupancy detection, physical activity detection, and general/personalized air quality monitoring), as well as up to 65% latency improvements with minimal accuracy loss. This exposes intermediate context for reuse, and the resulting systems can be extended and improved as new data becomes available.

Acknowledgment

This work is sponsored in part by TerraSwarm, a funded center of STARnet, a Semiconductor Research Corporation (SRC) program sponsored by MARCO and DARPA, as well as NSF Grant No. 1446912.

References

[1] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 414–454, 2014.

[2] J. C. Hammer and T. Yan, "Poster: A virtual sensing framework for mobile phones," in *Proc. of the 12th annual int. conf. on Mobile systems, applications, and services*. ACM, 2014, pp. 371–371.

[3] P. Rashidi, D. Cook, L. Holder, and M. Schmitter-Edgecombe, "Discovering activities to recognize and track in a smart environment," *IEEE Trans. on Knowledge and Data Eng.*, vol. 23, no. 4, pp. 527–539, 2011.

[4] C. LeRougea, J. Mab, S. Snehac, and K. Tolle, "User profiles and personas in the design and development of consumer health technologies," *Int. Jour. of Medical Informatics*, vol. 82, no. 11, pp. 251–268, 2013.

[5] N. Nikzad, N. Verma, C. Ziftci, E. Bales, N. Quick, P. Zappi, K. Patrick, S. Dasgupta, I. Krueger, T. Š. Rosing *et al.*, "Citisense: improving geospatial environmental assessment of air quality using a wireless personal exposure monitoring system," in *Proceedings of the conference on Wireless Health*. ACM, 2012, p. 11.

[6] A. Solanas, C. Patsakis, M. Conti, I. Vlachos, V. Ramos, F. Falcone, O. Postolache, P. Perez-martinez, R. Pietro, D. Perrea *et al.*, "Smart health: a context-aware health paradigm within smart cities," *Communications Magazine, IEEE*, vol. 52, no. 8, pp. 74–81, 2014.

[7] M. Rudary, S. Singh, and M. E. Pollack, "Adaptive cognitive orthotics: combining reinforcement learning and constraint-based temporal reasoning," in *Proc. of the 21st Int. conference on Machine Learning*.

[8] M. M. Baig and H. Gholamhosseini, "Smart health monitoring systems: an overview of design and modeling," *Journal of medical systems*, vol. 37, no. 2, pp. 1–14, 2013.

[9] Y.-J. Moon, Y.-H. Hwang, and S. Cho, *Advanced Multimedia and Ubiquitous Engineering*, ser. Lecture Notes in Electrical Engineering. SpringerLink, 2015, vol. 354, ch. An Empirical Study of Impacts of User Intention for Smart Wearable Devices and Use Behavior.

[10] M. E. Porter and J. E. Heppelmann, "How smart, connected products are transforming competition," *Harvard Business Review*, vol. 92, no. 11, pp. 64–88, 2014.

[11] "Smart Health IT - connecting health system data to innovators' apps," 2016.

[12] R. S. Weinstein, A. M. Lopez, B. A. Joseph, K. A. Erps, M. Holcomb, G. P. Barker, and E. A. Krupinski, "Telemedicine, telehealth, and mobile health applications that work: opportunities and barriers," *The American journal of medicine*, vol. 127, no. 3, pp. 183–187, 2014.

[13] J. Venkatesh, C. Chan, A. S. Akyurek, and T. S. Rosing, "A modular approach to context-aware iot applications," in *2016 IEEE Int. Conf. on Internet-of-Things Design and Implementation*, 2016, pp. 235–240.

[14] J.-H. Hong, S.-I. Yang, and S.-B. Cho, "Conamsn: A context-aware messenger using dynamic bayesian networks with wearable sensors," *Expert Systems with Applications*, vol. 37, no. 6, p. 46804686, 2010.

[15] S. Lee and K. C. Lee, "Context-prediction performance by a dynamic bayesian network: Emphasis on location prediction in ubiquitous decision support environment," *Expert Systems with Applications*, vol. 39, no. 5, p. 49084914, 2012.

[16] H. Chen, T. Finin, and A. Joshi, "An ontology for context-aware pervasive environments," *The Knowledge Engineering Review*, vol. 18, no. 3, pp. 197–207, 2004.

[17] S. K. Madhu, V. C. Raj, and R. M. Suresh, "An ontology-based framework for context-aware adaptive e-learning system," in *Proc. of the Int. Conf. on Computer Communication and Informatics (ICCI)*, 2013.

[18] Google now. [Online]. Available: http://www.google.com/landing/now

[19] S. Bandyopadhyay, M. Sengupta, S. Maiti, and D. Subhajit, "A survey of middleware for internet of things," *Recent Trends in Wireless and Mobile Networks*, 2011.

[20] T. Gu, X. Wang, H. Pung, and D. Zhang, "An ontology-based context model in intelligent environments," in *Proc. of communication networks and distributed systems modeling and simulation conf.*, 2004.

[21] C. Perera, A. Zaslavsky, M. Compton, P. Christen, and D. Georgakopoulos, "Context aware sensor configuration model for internet of things," in *Proc. of the 2013th Int. Conf. on Posters & Demonstrations Track-Volume 1035*, 2013, pp. 253–256.

[22] B. O. Akyurek, A. S. Akyurek, J. Kleissl, and T. S. Rosing, "Tesla: Taylor expanded solar analog forecasting," in *Smart Grid Communications, 2014 IEEE Int. Conf. on*, 2014, pp. 127–132.

[23] A. Battaglini, J. Lilliestam, A. Haas, and A. Patt, "Development of supersmart grids for a more efficient utilisation of electricity from renewable sources," *Journal of Cleaner Production*, 2009.

[24] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[25] A. Katasonov, O. Kaykova, O. Khriyenko, S. Nikitin, and V. Y. Terziyan, "Smart semantic middleware for the internet of things," in *Proc. of the 5th Int. Conf. on Informatics in Control, Automation and Robotics*, 2008.

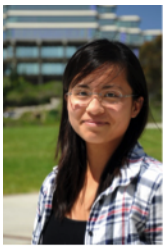
[26] D. Baron, S. Sarvotham, and R. G. Baraniuk, "Bayesian compressive sensing via belief propagation," *Signal Processing, IEEE Transactions on*, vol. 58, no. 1, pp. 269–280, 2010.

[27] J. Z. Kolter and M. J. Johnson, "Redd: A public data set for energy disaggregation research," in *Workshop on Data Mining Applications in Sustainability (SIGKDD), San Diego, CA*, vol. 25, 2011, pp. 59–62.

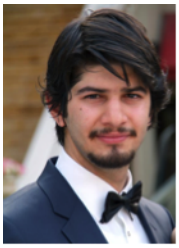
[28] K. Ellis, S. Godbole, S. Marshall, G. Lanckriet, J. Staudenmayer, and J. Kerr, "Identifying active travel behaviors in challenging environments using gps, accelerometers, and machine learning algorithms," *Frontiers in public health*, vol. 2, pp. 39–46, 2014.

Jaganathan Venkatesh is a PhD graduate in Computer Science and Engineering Department at UC San Diego. He was advised by Prof. Tajana Rosing as part of the System Energy Efficiency Lab (SEE-Lab). He received his Bachelor of Science degrees in Electrical Engineering and Computer Science from the University of Virginia, Charlottesville in June 2008. His research interests are energy efficiency, automation, and validation of renewable-energy systems and smart grids and power- and context-aware computing in embedded and mobile systems.

Baris Aksanli is an assistant professor at Electrical and Computer Engineering department of San Diego State University. He received his PhD and MS degrees in Computer Science from UC San Diego, and two BS degrees in Computer Engineering and Mathematics from Bogazici University, Turkey. His research interests include energy efficient cyber physical systems, human behavior modeling for the Internet of Things, big data for energy efficient large-scale systems. He won the Internet2 IDEA Award with his work in Lawrence Berkeley National Laboratory and Spontaneous Recognition Award from Intel.



Christine S. Chan is a PhD candidate in the Electrical and Computer Engineering Department at UC San Diego. She received her BS from the University of Illinois, Urbana-Champaign in 2011, and MS from UC San Diego in 2013, both in Computer Engineering. Her main research interest is in optimizing energy efficient systems according to both human and machine context. In particular, she is developing embedded devices for environmental sensing and user context extraction.



Alper Sinan Akyurek is a Postdoctoral Researcher in Computer Science and Engineering at UC San Diego. His current research is on context-aware control and optimization for energy efficiency in the smart grid. He obtained his Ph.D. (2017) from UCSD and, his M.Sc. (2011) and B.Sc. (2008) degrees in Electrical and Electronics Engineering from Middle East Technical University. He also worked as a Senior Design Engineer on Wireless Networks at Aselsan, Turkey.

Tajana Simunic Rosing is a Professor, a holder of the Fratamico Endowed Chair, and a director of System Energy Efficiency Lab at UCSD. She is currently heading the effort in SmartCities as a part of DARPA and industry funded TerraSwarm center. During 2009-2012 she led the energy efficient datacenters theme as a part of the MuSyC center. Her research interests are energy efficient computing, embedded and distributed systems. Prior to this she was a full time researcher at HP Labs while being leading research part-time at Stanford University. She finished her PhD in 2001 at Stanford University, concurrently with finishing her Masters in Engineering Management. Her PhD topic was Dynamic Management of Power Consumption. Prior to pursuing the PhD, she worked as a Senior Design Engineer at Altera Corporation.