2014 IEEE International Conference on Internet of Things (iThings 2014), Green Computing and Communications (GreenCom 2014), and Cyber-Physical-Social Computing (CPSCom 2014)

Estimating Contextual Situations using Indicators from Smartphone Sensor Values

Stefan Forsström and Victor Kardeby
Department of Information and Communication Systems
Mid Sweden University, Sundsvall SE-85170, Sweden
Email: stefan.forsstrom@miun.se and victor.kardeby@miun.se

***Abstract*—Current context-aware applications often use the location of a user as the only indication of the current situation. These existing applications are therefore limited in their situation awareness, because of the poor indoor resolution of the location sensor and its high resource consumption. In response to these limitations we present an approach to estimate the contextual situation of a user without using resource inefficient location sensors. Our proposed solution utilizes a wide range of low powered sensors, together with two modified machine learning techniques to estimate the situation in a more resource efficient manner. Simulations and a proof-of-concept application show that the situation of a user can be determined within 50 ms at an accuracy above 90%, when only using the low energy sensors available on a smartphone and its limited processing power.**

I. Introduction

Today we see a large proliferation in situation-aware applications, which can change their behavior depending on the situation of their user. These applications are called context-aware, because they are made aware of their surroundings through the use of a large amount of collaborating sensors. This idea of context awareness have been studied for quite some time, such as by Dey and Abowd [1]. But previous work mostly focused on quite limited scenarios or narrow use cases such as active badges [2] and cyberguides [3]. It is only recently that such knowledge is applied to more general cases and beyond the scope of purely location-aware applications [4], [5], [6]. This project will approach the research area from the point of view that there will be a single widespread Internet-of-Things in the future [7], where all objects will be connected and sharing sensor information with each other. However, each connected sensor will still have limited individual resources in the form of battery and computational power. Hence, there is a need to utilize their collective collaborative resources in order to together create more powerful context-aware applications.

Simple context-aware applications utilize sensors to achieve reactive behavior. One simple example is to change the layout on the screen as the device orientation changes, using the accelerometer sensor. Traditional location sensors such as the Global Positioning System (GPS), are a powerful tool in creating more complex types of context-aware applications because of their strong indicator of the physical location. But there are situations where this type of location sensor is suboptimal because of its significant battery consumption and lack of accuracy in underground or indoor locations [8]. Therefore, this paper will focus on the ability to estimate the situation of a person, when the situation can not solely be determined by the location of a person. For example, a home can be seen as a single location that can have many different situations, such as watching TV, eating dinner, cleaning, sleeping, etc. Our idea is to estimate the situation of a person using multiple different weak indicators and low energy sensors, i.e. a form of situation estimation [6] using sensor fusion [9].

In this paper, we concentrate on a sensor platform which people carry with them most of the time, namely today's smartphones. Hence only a limited set of available sensor information will be used from the smartphone to avoid excessive resource consumption. The GPS sensor will be avoided because it consumes too much battery for continuous usage over an extended period of time. The idea of using smartphone sensors for estimating certain behavior have been successfully applied in different specific and quite limited scenarios, such as driving behavior [10] and places of interest [11]. Hence, this paper investigates if smartphones could be used to perform situation estimation in a more general case, by using sensor values that only provide weak indications of the situation. All this in order to provide a better service, end user experience, optimized communication, and longevity for context-aware applications on the Internet-of-Things. To address the previously stated problem on estimating the contextual situation using limited resources, we have determined four concrete requirements which must be fulfilled by a proposed solution. These are in detail:

1) It must be able to estimate the situation in less than 100 ms, in order to support real-time applications which require continuous evaluation of the situation. For example, an application might want to use the system to detect sudden sensor value changes in order to avoid otherwise unpredictable accidents.
2) It must be reliable and return answers which the application and user can act reliably upon (more than 90% accuracy). For example, an application might want to control different actuators depending on the situation of the user, such as controlling the heating of a house.
3) It must be able run the situation estimation on limited hardware and with low resource consumption. This because an application might want to use the solution on the Internet-of-Things, were there will be a large amount of resource limited devices, including mobile phones and wireless sensor network motes.

The rest of the paper is structured as follows: In section II we present our approach to estimating contextual situations. In section III we present a simulation performed to validate the proposed algorithm. Section IV resents the simulation

978-1-4799-5967-9/14 $31.00 © 2014 IEEE
DOI 10.1109/iThings.2014.43

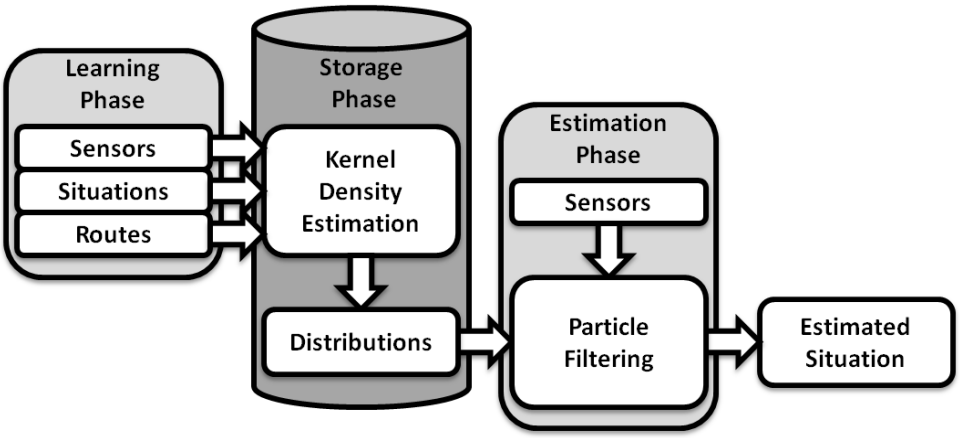

Fig. 1. An overview of the approach and architecture

measurements and section V presents a proof-of-concept implementation of the algorithm as a mobile application. Section VI highlights the results of the research and section VII presents possible future work. Finally, section VIII present the conclusions drawn from this research.

II. Approach

Our approach involves combining two specific well used machine learning techniques [12], namely particle filtering [13], [14] and kernel density estimation [15]. This in order to estimate the situation of a person when only given a set of the person's current sensor values. Kernel density estimation is a method to estimate the probability density function of an unknown random variable using a finite set of Independent and Identically Distributed (IID) samples. Basically, kernel density estimation can estimate the unknown underlying distribution when given a set of values. It achieves this by using a kernel function and a smoothing parameter, which in this approach will be the standard normal density function. In the end, it will be used to estimate the unknown distribution of the sensor values for a specific situation. Particle filtering is a Monte Carlo Markov Chain (MCMC) method that can estimate a Bayesian model using sequential simulation, typically estimating different types of hidden Markov models where an exact inference is not possible. The estimation use a large number of particles that traverses the model and tries to find which particle that is most probable at the correct state. The kernel density estimator and particle filter will be used to estimate the situation of a person, given all available situations, the transitions between situations, and how the sensor values previously have been distributed when someone was in the same situation. But to ensure proper performance of both the kernel density estimator and particle filter, they require a sufficient amount of collected and prerecorded sensor values. With these collected, the probability density distributions can be used in the particle filtering algorithm to estimate the current situation of a person.

A. Architecture

Figure 1 shows an overview of our approach and architecture to solve the problem. The architecture is divided into three phases, namely learning phase, persistent storage phase, and estimation phase. The general idea is that the learning phase record situations and their corresponding sensor values, the storage phase stores these values and optimizes them for later usage, and the estimation phase retrieves the sensor distributions from the storage and performs the estimation. Our solution differs from related work on certain key aspects, such as no reduction in dimensionality, the use of only raw sensor values, and no preprocessing of the data. But the main difference is in our focus on low cost sensors and simple computations. Many related approaches use strong indicators to estimate the situation, especially the precise location. Our approach is for example similar to [16], but we do not rely on location as a strong indicator. The approach is also similar to [17] but without expensive feature extraction. Furthermore, we focus on the sensors which people today carry around with them all the time, the sensors available on today's smartphones.

In our architecture, the learning phase is used to record the available situations and their corresponding raw sensor values. This is done by querying the user about their current situation and record the sensor values for as long as the user claim they are in a particular situation. We also record the transitions between situations, to determine the routes and relations between situations. All the raw sensor values, their corresponding situations, and the routes between situations are recorded and sent to the persistent storage for later usage. In the storage phase the recorded information is optimized if the volume of sensor values are too large, this in order to limit the amount of data to be sent to the end devices in the next phase. The reduction is done by applying kernel density estimation on the stored sensor values and resample a fixed set of similarly distributed values. This fixed set is continuously updated and stored for later usage when users want to estimate their situations. In the estimation phase a device downloads the latest data from the storage, which will be used as the model of the world. The device senses its current sensor values and applies them to the model using kernel density estimation to get the estimated distributions and particle filtering to estimate the most probable situation.

B. Kernel Density Estimation

In order to estimate the probability density function of a sensor at a specific situation we must have previously sampled a number of sensor values $(x_1, x_2, ..., x_n)$. The kernel density estimation is then expressed as:

$$f(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x - x_i}{h}) \tag{1}$$

where $K(u)$ is the selected kernel function and $h > 0$ is a smoothing parameter. We use the standard normal density function, which is a commonly used kernel function:

$$K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2} \tag{2}$$

Selecting the optimal kernel function is an ongoing research topic in its own, and it is therefore deferred as future work. For example, [18] proposed a hybrid selection method and a [19] propose a variable density kernel. In our solution, the smoothing parameter h is estimated for Gaussian based functions [20] with:

$$h = (\frac{4\sigma^5}{3n})^{\frac{1}{5}} \tag{3}$$

Where σ is the standard deviation of the samples. Thus, the method for calculating the probability density P for a chosen value x is then expressed in the following way:

$$P = \frac{1}{nh\sqrt{2\pi}} \sum_{i=0}^{n} e^{-\frac{1}{2}(\frac{x-x_i}{h})^2} \tag{4}$$

C. Particle Filtering

A modified particle filter technique is used for situation estimation. The estimation is done using a large number of particles that traverses a model of the situations and tries to find which particle that is most probable at the correct situation. Particle filtering have been utilized in many similar applications, for example multi objects tracking in sports [21], robotics [22], and vehicle navigation [23]. The major differences from a regular particle filter and our proposed method, is the particle weight calculations from sensor values and the modified resampling method. In our method, a weighting for each sensor w^{sensor} is calculated using its current sensor value in the distribution d_{value} for the guessed situation in the model m. The weighting for the guessed situation $w_t^{situation}$ at time t is calculated by multiplying all available sensor weights for that situation with the weight from the previous run of the algorithm. Thus, the weight is calculated with our modified weight formulas:

$$w_t^{sensor} = p(d_{value}|m) \tag{5}$$

$$w_t^{situation} \propto w_{t-1}^{situation} * \prod w_t^{sensor} \tag{6}$$

An overview of the complete modified particle filtering algorithm can be seen in the following algorithm, where the modified resampling method can also be seen.

Require: $m = \{(z_1, \{d_z\}), (z_2, \{d_z\}), ..., (z_n, \{d_z\})\}$
Require: S, the total number of particles
Require: s_{min}, the minimum efficient sample size
Require: n_{max}, the maximum number of iterations
loop
  for all particles s at time t **do**
    Draw a situation $z_t^s \sim q(z_t|z_{t-1}^s, m)$
    for all sensors i **do**
      Compute sensor weight $w_t^i = p(d_z^i|m)$
    end for
    Compute particle weight $w_t^s \propto w_{t-1}^s * \prod_{i=1}^{S} w_t^i$
  end for
  Normalize all weights $w_t^s = \frac{w_t^s}{\sum_{s'} w_t^{s'}}$
  Compute $S_{eff} = \frac{1}{\sum_s (w_t^s)^2}$
  if $S_{eff} > S_{best}$ **then**
    $z_{best} = \max w_t^s$
    $S_{best} = S_{eff}$
  end if
  if $S_{eff} < S_{min}$ and $n_{iteration} < n_{max}$ **then**
    $n_{iteration} + 1$
    for all particles s indexed $0 \rightarrow S/2$ **do**
      Resample $z_t^s \sim q(w_t^z)$
      $w_t^s = 1/S$
    end for
    for all particles s indexed $S/2 \rightarrow S$ **do**
      Resample $z_t^s \sim q(z_t)$
      $w_t^s = 1/S$
    end for
  else
    return z_{best}, the best estimated situation
  end if
end loop

The order in the algorithm is that each particle will sample a possible situation change and calculate the average weight using the sensor values for that situation based on the previous situation. The weights are then normalized to calculate the estimated effective sample size. If this effective sample size is below a selected threshold it is resampled and the algorithm restarts. The resampling is done by redistributing the particles according to the current situation distribution. When the effective sample size is above the selected threshold or when the number of resamples exceeds a selected threshold we find our estimated situation by finding the particle with the highest weight from the iteration with the highest effective sample size.

In detail, the algorithm require knowledge of the model m which contains the possible situations z and the sensor values distributions at that situation d_z. Each particle s will draw a random possible situation z using the proposal distribution function q at the current time t, namely z_t^s. This is calculated given previous situation of the particle at time $t-1$ and the possible situation changes from the model m. Then each sensor i will calculate its current weight w, namely w_t^i. This is done by using the distribution of the sensor d_z^i at the current situation and the model. After each sensor have been weighted, the total particle weight for the situation w_t^s is calculated. The total particle weight is proportional to the product of all the particle's current sensor weights w_t^i, times the previous weight of the particle w_{t-1}^s. When all particles have been weighted they need to be normalized, which is done by dividing the

current weight w_t^s with the total particle weight given by all the particles $w_t^{s'}$. The effective sample size S_{eff} is calculated and compared to the threshold effective sample size S_{min}, and if S_{eff} is less than S_{min} the particles will be resampled. The resampling of particles is done in two steps. First, half of the particles resample their situation from the current calculated weight distribution of the situations w_t^z in order to concentrate particles around situations with high weight. Secondly, the other half of the particles randomly resample from all available situations regardless of the situation weights, this in order to not completely focus all particles to a small subset of the situations. Finally the particle weights are reset to $1/S$, where S is the total number of particles.

D. Summary

To summarize, users will need to record the possible situations, sensor values for each of the situations, and the routes between situations. These will be optimized in a storage system using kernel density estimation, to limit the required communication to the end device. In the end device, both kernel density estimation and particle filtering will be used on the current sensor values to estimate the current situation. The maximum required calculations required for the situation estimation c_{max} is expressed in equation 7 where S is the number of particles, i is the number of sensors, n_{kde} is the number of sampled values used in the kernel density estimator, and n_{max} is the maximum number of resamples.

$$c_{max} = S * (i * 3n_{kde} + n_{max}) \tag{7}$$

III. Simulation

We have created a simulation environment to test and evaluate our algorithm. The purposes of this evaluation is to investigate if the algorithm can successfully estimate the situation from a number of weak sensor based indicators. The methodology of the evaluation will be to first simulate a training set of sensor values for a number of different situations, to then apply our algorithm to validate that we can successfully identify the situation. The setup of the simulation environment follows the approach shown in figure 1 in section II, where there are three main components. One teacher components that generates 300 sensor values for each sensor type and location to teach the system the possible situations, routes between the situations, and example sensor values at the situations. The second component is a simple server which stores these sensor values and creates the distributions from them, using the kernel density estimation, reducing the data set to 200 values. The third component is the estimator component which retrieves knowledge about the situations and their sensor value distributions from the server. Estimations are then made locally on the end device, given the current sensor values. The simulation environment also makes it possible to tweak all the input parameters to the particle filter, for example the number of particles, efficient sample size, and maximum number of allowed resample iterations. The procedure used in our measurements will be to first generate the learning set of data, then run the simulation on the data, and finally measure the accuracy and for how long time the algorithm was running. The measurements will the repeated for two different setups, one on a computer and a second one on a smartphone.

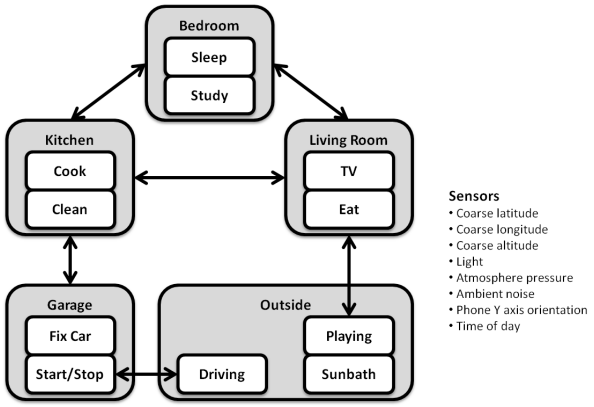

Fig. 2. An overview of the world used in the simulation

The simulation emulates a use-case scenario of a home environment. The sensors and situations will be created in such a way that it should reflect how the proposed system should work when applied to a real life situation with actual sensors. The possible situations and sensors have been chosen to be relevant and appropriate for a person to switch between in a home environment. In detail, the possible situations in this scenario are: Cooking food, cleaning the kitchen, eating food, watching TV, sleeping, studying, playing outside, sunbathing, fixing the car, starting/stopping the car, and lastly driving the car. Furthermore, not all situations can change between each other, for example to drive car the entity must first go through start/stop car. In our simulated scenario we will use the following sensors: Coarse location (an inaccurate and low cost location estimation using the radio base stations), coarse altitude, light, atmosphere pressure, ambient noise, phone y axis orientation, and time of day. All of the sensors will be treated as continuous values, even though some are more discrete in nature. See figure 2 on how the world is viewed in this scenario and a list of the available sensors that are being simulated. But it is important to understand that even if our simulation takes coarse location into consideration, it is not prioritized differently from any other sensor. Furthermore, we only rely on low accuracy and low cost location sensing such as the network based location on the smartphone.

In this simulation the values of the sensors are generated from different types of random distribution functions feeding into the kernel density estimation component. Which will in turn create an estimation of the underlying distributions and make it possible for the particles to calculate sensor weights depending on their drawn situations. In summary, we have selected the generated sensor value distributions so they would emulate real-world values. For example, when a person cook food they are always centered in the kitchen which has certain latitude/longitude with a deviation of the values to simulate the imprecision of actual sensor values. Also, when we are cooking the phone's orientation sensor is also almost always indicating that we are standing up. Furthermore, cooking can also only change situation to either cook, clean, eat, TV, sleep, study, fix-car, or start/stop. This because we are inside the kitchen which is connected to the other rooms in the house, and in this case the kitchen has a door to the garage. Table I, II, and III, shows the generator functions we used for the sensor

```
============================================================
New Best: Clean at sEff: 1.0000000002399547 from run nr 0
New Best: Clean at sEff: 2.000000000719864 from run nr 1
I gave up after 3 resamples...
sEff: 2.000000000719864
EstimatedSituation: Clean
ActualSituation: Clean(2)
----STATISTICS----
Accuracy : 0.6163232646529306
ResampleFrequency : 2.9656931386277257
Time: 12628430ns => 12.62843ms
AvgTime: 1.3296710084616924E7ns => 13.296710084616922ms
Stdev: 1291370.292713293ns => 1.291370292713293ms
============================================================
```

Fig. 3. An example of the output from the simulator

values in each situation. N denotes a normal distribution with an average and variance, DUC denotes a discrete uniform distribution with categorical values.

IV. Simulation Measurements

The results from this simulation indicate that the modified particle filter approach works well in combination with the kernel density estimation, in order to fast and efficiently estimate the situation of a person. When only given its current sensor values and the sensor value distributions in all the different possible situations as prior knowledge. In detail, the particle filter itself only need four input variables, the amount of particles, the sought after minimum efficient sample size S_{min}, the maximum amount of allowed resample iterations n_{max}, and the model world with the sensor distributions. An example of the output from the simulator can be seen in figure 3. This example show the actual situation, the estimated situation, the particle filters best efficient sample size so far, and some statistics such as average accuracy, average resample frequency, and measured time for the calculations. From this output we can determine the needed results to satisfy requirement 1 and 2, namely fast response and high accuracy.

A. Accuracy measurements

The measurements was run in 9 different setups with different amounts of particles, different sought after minimum efficient sample size S_{min}, and different maximum number of allowed resample iterations. In detail we have tested: 10 particles with an S_{min} of 3, 6, and 9, with three maximum allowed resample iterations. 30 particles with an S_{min} of 9, 18, and 20, with six maximum iterations. And finally 50 particles with S_{min} 15, 30, and 45, with ten maximum iterations. Furthermore, each test case was run consequent 10 000 steps. This in order to see how these three input variables effected the accuracy, resampling rate, and computational time.

Tables IV to VI shows the acquired results when we ran the simulator. The accuracy is the number of correct estimation divided by the total number of estimations. The resample frequency is the average number of resamples needed per step, before finding a satisfactory answer or exceeding the allowed maximum number of resampling iterations. And the response time is the measured time to perform a single estimation, shown as an average with its standard deviation. In detail, it is shown that at as early as 30 particles and a S_{min} of 6 there is an accuracy of 89%, without any severe impact on the computational time (35ms on average). Additionally it shows

TABLE IV
Results for 10 particles

Particles/S_{min}/n_{max}	10/2/3	10/4/3	10/6/3
Accuracy	0.5940	0.6161	0.6164
Resample frequency	0.9291	2.553	2.965
Mean time	2.172 ms	2.629 ms	2.707 ms
Time stdev	0.6437 ms	0.6260 ms	0.5797 ms

TABLE V
Results for 30 particles

Particles/S_{min}/n_{max}	30/6/6	30/12/6	30/18/6
Accuracy	0.8971	0.9158	0.9172
Resample frequency	2.580	5.477	5.923
Mean time	3.213 ms	3.688 ms	3.745 ms
Time stdev	0.7740 ms	0.7553 ms	0.6639 ms

TABLE VI
Results for 50 particles

Particles/S_{min}/n_{max}	50/10/10	50/20/10	50/30/10
Accuracy	0.9454	0.9607	0.9654
Resample frequency	3.892	9.059	9.916
Mean time	3.919 ms	4.741 ms	4.764 ms
Time stdev	1.156 ms	1.156 ms	0.9772 ms

TABLE VII
Results on the Android smartphone

Particles/S_{min}/n_{max}	10/4/3	30/12/6	50/20/10
Accuracy	0.6152	0.9147	0.9692
Resample frequency	2.516	5.533	9.089
Mean Time	16.08 ms	38.44 ms	82.17 ms
Stdev Time	5.650 ms	12.34 ms	36.13 ms

that if the desired efficient sample size is increased, there is an increase in the computational time but not the same gain in accuracy. Therefore, the amount of particles seem to have a better impact on the accuracy than the S_{min}.

B. Limited device measurements

Goal 3 was to inspect if our method for determining context situations can be used on limited devices such as smartphones. Therefore one test runs the same estimator code on an Android smartphone (a Samsung Galaxy Nexus). Table VII show the same simulation results for the version running on the smartphone, for the same settings. From this table we can see that even on a smartphone, it is possible to effectively determine the contextual situation within limited time bounds. This is a very interesting result in relation to requirement 3, because it indicates that the particle filtering technique can be applied for real-time context-aware applications on limited mobile devices.

V. Proof-of-Concept Application

The results from the simulation verify that the approach is valid in a simulation environment. However this does not conclude that the method will operate as accurately in a real-world deployment. For example, real-world sensor values may not be independent and identically distributed, they may contain anomalies, and the recording of contextual situations

TABLE I
GENERATOR FUNCTIONS FOR COOK, CLEAN, SLEEP, AND STUDY

Situation	Cook	Clean	Sleep	Study
Coarse latitude	$N(62.394, 0.001^2)$	$N(62.394, 0.001^2)$	$N(62.398, 0.001^2)$	$N(62.398, 0.001^2)$
Coarse longitude	$N(17.289, 0.0005^2)$	$N(17.289, 0.0005^2)$	$N(17.288, 0.001^2)$	$N(17.288, 0.001^2)$
Coarse altitude	$N(100, 1^2)$	$N(100, 1^2)$	$N(100, 1^2)$	$N(100, 1^2)$
Light	$N(200, 20^2)$	$N(200, 20^2)$	$N(20, 5^2)$	$N(100, 20^2)$
Sound	$N(60, 20^2)$	$N(80, 20^2)$	$N(30, 10^2)$	$N(50, 10^2)$
Atmosphere	$N(1007, 1^2)$	$N(1007, 1^2)$	$N(1007, 1^2)$	$N(1007, 1^2)$
Time	$DUC(16-19)$	$DUC(19-20)$	$DUC(22-23, 0-8)$	$DUC(18-20)$
Phone orient.Y	$N(90, 10^2)$	$N(90, 10^2)$	$N(0, 5^2)$	$N(0, 10^2)$

TABLE II
GENERATOR FUNCTIONS FOR EAT, FIXCAR, AND STARTSTOPCAR

Situation	Eat	FixCar	StartStopCar
Coarse latitude	$N(62.396, 0.002^2)$	$N(62.391, 0.0005^2)$	$N(62.391, 0.0005^2)$
Coarse longitude	$N(17.287, 0.0005^2)$	$N(17.289, 0.0005^2)$	$N(17.289, 0.0005^2)$
Coarse altitude	$N(100, 1^2)$	$N(100, 1^2)$	$N(100, 1^2)$
Light	$N(200, 20^2)$	$N(100, 20^2)$	$N(100, 20^2)$
Sound	$N(70, 10^2)$	$N(50, 10^2)$	$N(60, 10^2)$
Atmosphere	$N(1007, 1^2)$	$N(1007, 1^2)$	$N(1007, 1^2)$
Time	$DUC(7, 8, 12, 13, 18, 19)$	$DUC(20-22)$	$DUC(8, 9, 16, 17)$
Phone orient.Y	$N(90, 10^2)$	$N(0, 10^2)$	$N(90, 10^2)$

TABLE III
GENERATOR FUNCTIONS FOR PLAYING, SUNBATHING, DRIVING, AND TV

Situation	Playing	Sunbathing	Driving	TV
Coarse latitude	$N(62.395, 0.0025^2)$	$N(62.395, 0.0025^2)$	$N(62.395, 0.01^2)$	$N(62.396, 0.002^2)$
Coarse longitude	$N(17.283, 0.0015^2)$	$N(17.283, 0.0015^2)$	$N(17.285, 0.01^2)$	$N(17.287, 0.0005^2)$
Coarse altitude	$N(100, 1^2)$	$N(100, 1^2)$	$N(100, 10^2)$	$N(100, 1^2)$
Light	$N(2000, 200^2)$	$N(2200, 200^2)$	$N(1000, 200^2)$	$N(300, 20^2)$
Sound	$N(70, 20^2)$	$N(50, 20^2)$	$N(60, 10^2)$	$N(80, 30^2)$
Atmosphere	$N(1007, 1^2)$	$N(1007, 1^2)$	$N(1007, 1^2)$	$N(1007, 1^2)$
Time	$DUC(10-20)$	$DUC(10-14)$	$DUC(8, 9, 16-17)$	$DUC(17-23)$
Phone orient.Y	$N(90, 20^2)$	$N(0, 10^2)$	$N(90, 10^2)$	$N(0, 10^2)$

is not trivial. Because of this, we have built a proof-of-concept mobile application which utilizes the presented method. The application can create new situations, record real world sensor values to these situations, and finally estimate the contextual situation in real-time from its current sensor values. Figure 4 shows four screenshots from the proof-of-concept application. The first screenshot show the build mode, in which the situation model is created. The transitions between situations are set up using a drag and drop user interface. The second screenshot shows the learn mode, where the user can teach the system sensor values corresponding to a particular situation. The third screenshot show the estimation mode, in which the application utilizes the created system and the learned sensor values to estimate the current situation based on the phone's latest sensor values. Lastly, the fourth screenshot show a settings screen in which different settings can be altered. It is difficult to evaluate this proof-of-concept in terms of accuracy because of the difficulties in precisely recording real world situations multiple times. However, it is apparent that the application is capable of correctly identifying the correct situation from a set of recorded situations. Such as distinguishing running from walking, etc. Furthermore, even in this resource constrained environment there is enough computational power for the device to do the computations and deliver estimations at an acceptable rate. But the general result of this proof-of-concept applications is that it is possible to provide context estimations using kernel density estimation and particle filtering techniques using real world sensor values on a smartphone.

The typical scenarios where we believe this system can easily be implemented are in different personal applications or to utilize it as an indicator to improve other applications. One example is to perform home automation, for example make music follow you around in the house and make the volume of the music change depending if you are cooking food or reading a book. We believe that it can be used to improve the accuracy of different mobility solutions for communication, by detecting an impending movement of the user. Because our system is so resource efficient it can also be utilized in resource limited devices, such as sensor motes in wireless sensor networks. In wireless sensor networks the solution can be used to estimate the situation of a specific sensor mote to make it react to changes in its surroundings, such as topology and environmental changes. Furthermore, because our system is good at detecting regular common situations it can also be used to detect an unknown or uncommon situations. This is interesting for eldercare, in order to identify when something does not correspond to an elder person's normal routine and hence an accident might have happened.

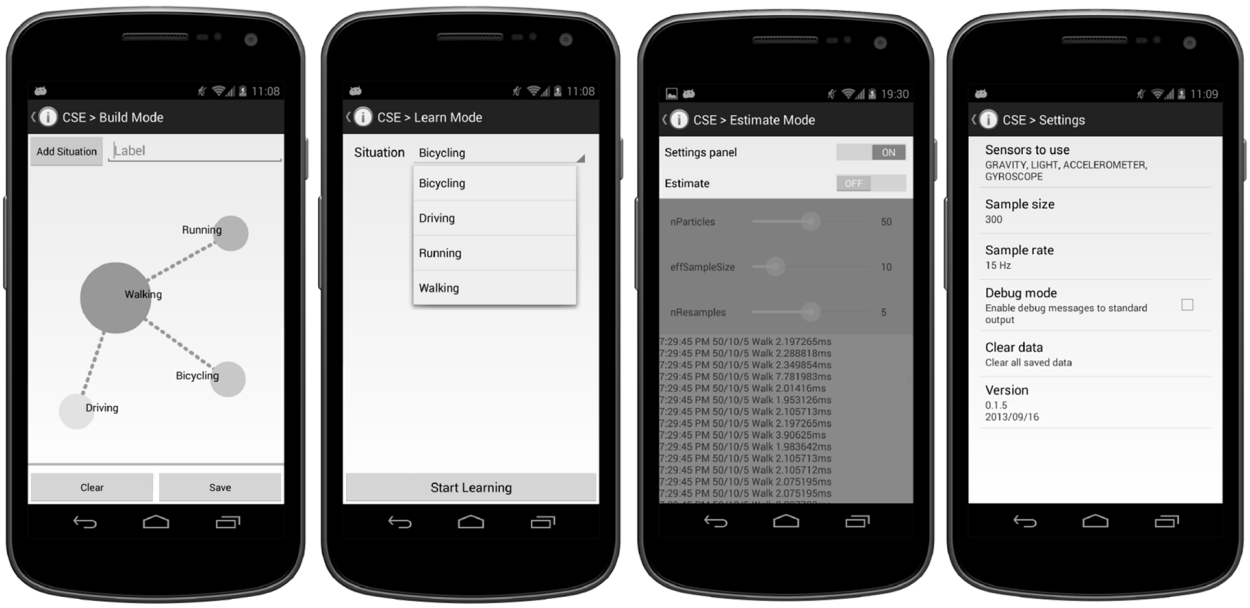

Fig. 4. Screenshots of the proof-of-concept application.

VI. Results

By applying the method presented in this publication we were able to find the most probable situation, given only the temporal, coarse, and unreliable sensor values from the user's smartphone. Without the use of expensive GPS location sensors, which is what related work such as [11] and [16] heavily relies on. The approach have also been evaluated on a device with limited resources in a home scenario, where it was observed that the particle filtering and kernel density estimation methods executes quickly even on limited hardware. In table VII we can see that the proposed solution satisfies requirement 1 on fast estimation, 2 on reliable answers, and 3 on limited hardware. Since, by using 30 particles, an S_{min} of 12, with a maximum resample threshold of 6, we achieve a 91.47% accuracy with only a 38.44 ms computation time on a limited smartphone device. We can also deduce that our solution should become more accurate as more people contribute and learn the system. This means that the estimation should become even better in the future, as both new sensor types and more sensor values are integrated.

VII. Future Work

Our current future work is focused on gathering sufficient data in order to evaluate the system in a large scale field trial with many different users and situations. The current scenario revolves around a person's home, which is a good verification model for a situation where a precision location sensor cannot be utilized. But as a useful application area, it is quite simple in its structure. We need to evaluate the system on more of the typical Internet-of-Things scenarios, for example logistics, public transportation, health-care, and personal safety. All which have higher demands, especially in the terms of scalability and the amount of connected devices. It would also be interesting to release the system to the public, for example by publishing the proof-of-concept application for anyone to download. This in order to have a large number of people contributing with their sensor values to identify a very wide spectrum of situations. Thus teaching the system to recognize any type of generic situation, regardless of scenario.

Furthermore, we are continuously tweaking the situation estimation algorithm, for example the amount of particles, efficient sample size, resample iterations, and sensor weighting. To find the optimal efficient sample size is paramount to minimize the calculation time of the algorithm. However, the optimal efficient sample size heavily relies on the amount of gathered sensor information and the scenario which to estimate the situation in. We are also exploring the possibility to use our system to optimize other components. By knowing the situation of a person, it is for example possible to predict future movement and behavior. To then perform proactive actions, such as prefetching certain information and perform intelligent caching in advance. We will explore the possibility to use this in scenarios with very high mobility, where a person's connectivity might be sporadic but predictable based on the situation.

VIII. Conclusion

The problem that was investigated in this paper was to estimate the situation of a person, where the situation can not solely be determined by the location. The proposed solution utilized kernel density estimations to create sensor value distributions for different situation, to then apply a modified particle filtering algorithm on the sensor values distributions to estimate the current situation. By applying this method we were able to find the most probable situation, given only the temporal, coarse, and unreliable sensor values from the user's smartphone. Thus, we can conclude that by using the contributions in this publication it is possible to the estimation of contextual situations on limited hardware, without the use of resource inefficient location sensors.

REFERENCES

[1] A. Dey and G. Abowd, "Towards a better understanding of context and context-awareness," in *CHI 2000 workshop on the what, who, where, when, and how of context-awareness*, 2000, pp. 304–307.

[2] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," *ACM Transactions on Information Systems (TOIS)*, vol. 10, no. 1, p. 102, 1992.

[3] G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton, "Cyberguide: A mobile context-aware tour guide," *Wireless networks*, vol. 3, no. 5, pp. 421–433, 1997.

[4] A. Schmidt, M. Beigl, and H. Gellersen, "There is more to context than location," *Computers & Graphics*, vol. 23, no. 6, pp. 893–901, 1999.

[5] J. Coutaz, J. L. Crowley, S. Dobson, and D. Garlan, "Context is key," *Communications of the ACM*, vol. 48, no. 3, pp. 49–53, 2005.

[6] J. Ye, S. Dobson, and S. McKeever, "Situation identification techniques in pervasive computing: A review," *Pervasive and Mobile Computing*, vol. 8, no. 1, pp. 36–66, 2012.

[7] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[8] M. B. Kjærgaard, H. Blunck, T. Godsk, T. Toftkjær, D. L. Christensen, and K. Grønbæk, "Indoor positioning using gps revisited," in *Pervasive Computing*. Springer, 2010, pp. 38–56.

[9] N. Bicocchi, G. Castelli, M. Mamei, and F. Zambonelli, "Improving situation recognition via commonsense sensor fusion," in *Database and Expert Systems Applications (DEXA), 2011 22nd International Workshop on*. IEEE, 2011, pp. 272–276.

[10] H. Eren, S. Makinist, E. Akin, and A. Yilmaz, "Estimating driving behavior by a smartphone," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 2012, pp. 234–239.

[11] R. Montoliu, J. Blom, and D. Gatica-Perez, "Discovering places of interest in everyday life from smartphone data," *Multimedia Tools and Applications*, vol. 62, no. 1, pp. 179–207, 2013.

[12] K. Murphy, *Machine Learning: a Probabilistic Perspective*. MIT press, 2012.

[13] A. Doucet, N. De Freitas, N. Gordon *et al.*, *Sequential Monte Carlo methods in practice*. Springer New York, 2001, vol. 1.

[14] M. Sanjeev Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *Signal Processing, IEEE Transactions on*, vol. 50, no. 2, pp. 174–188, 2002.

[15] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *The Annals of Mathematical Statistics*, vol. 27, no. 3, pp. 832–837, 1956.

[16] G. Tanaka and H. Mineno, "A method of estimating outdoor situation for lifelog generation," in *Consumer Electronics (GCCE), 2013 IEEE 2nd Global Conference on*. IEEE, 2013, pp. 361–362.

[17] S. A. Hoseini-Tabatabaei, A. Gluhak, and R. Tafazolli, "A survey on smartphone-based systems for opportunistic user context recognition," *ACM Computing Surveys (CSUR)*, vol. 45, no. 3, p. 27, 2013.

[18] M. Jiang and S. B. Provost, "A hybrid bandwidth selection methodology for kernel density estimation," *Journal of Statistical Computation and Simulation*, vol. 84, no. 3, pp. 614–627, Mar. 2014.

[19] Z. Zhang and Y. Zhang, "Variable kernel density estimation based robust regression and its applications," *Neurocomputing*, vol. 134, pp. 30–37, Jun. 2014.

[20] B. W. Silverman, *Density estimation for statistics and data analysis*. CRC press, 1986, vol. 26.

[21] K. Okuma, A. Taleghani, N. De Freitas, J. J. Little, and D. G. Lowe, "A boosted particle filter: Multitarget detection and tracking," in *Computer Vision-ECCV 2004*. Springer, 2004, pp. 28–39.

[22] G. G. Rigatos, "Extended kalman and particle filtering for sensor fusion in motion control of mobile robots," *Mathematics and computers in simulation*, vol. 81, no. 3, pp. 590–607, 2010.

[23] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle filters for positioning, navigation, and tracking," *Signal Processing, IEEE Transactions on*, vol. 50, no. 2, pp. 425–437, 2002.