Towards Enabling Probabilistic Databases for Participatory Sensing

Nguyen Quoc Viet Hung #1, Saket Sathe *2, Duong Chi Thang #3, Karl Aberer #4

École Polytechnique Fédérale de Lausanne
1 quocviethung.nguyen@epfl.ch
3 thang.duong@epfl.ch
4 karl.aberer@epfl.ch

* *IBM Melbourne Research Laboratory*
2 ssathe@au.ibm.com

***Abstract*—Participatory sensing has emerged as a new data collection paradigm, in which humans use their own devices (cell phone accelerometers, cameras, etc.) as sensors. This paradigm enables to collect a huge amount of data from the crowd for world-wide applications, without spending cost to buy dedicated sensors. Despite of this benefit, the data collected from human sensors are inherently uncertain due to no quality guarantee from the participants. Moreover, the participatory sensing data are time series that not only exhibit highly irregular dependencies on time, but also vary from sensor to sensor. To overcome these issues, we study in this paper the problem of creating probabilistic data from given (uncertain) time series collected by participatory sensors. We approach the problem in two steps. In the first step, we generate probabilistic times series from raw time series using a dynamical model from the time series literature. In the second step, we combine probabilistic time series from multiple sensors based on the mutual relationship between the reliability of the sensors and the quality of their data. Through extensive experimentation, we demonstrate the efficiency of our approach on both real data and synthetic data.**

I. Introduction

Participatory sensing, where participants proactively report their observations, has emerged as an important data collection paradigm. In this paradigm, human acts as the sensors or employs their own devices to perform sensing tasks. Examples of these sensors include cell phone accelerometers, cameras, and GPS devices. As the number of sensors owned by an individual increases, participatory sensing is becoming popular. The potential of participatory sensing is tremendous as it can harness the wisdom of the crowd to collect data for wide-ranging applications such as geotagging, environmental monitoring, and public health. These applications would be financially infeasible if traditional approaches using dedicated sensors are applied. Although applications of participatory sensing are abundant, it faces a serious problem from the inferior quality of collected data. Data collected from individual participants and their devices are inherently uncertain due to low sensor quality, unstable communication channels or even intent to cheat the system.

One of the most effective ways to deal with uncertain data is to employ probabilistic approaches. In recent years there have been a plethora of methods for managing uncertain data [1], [2], [3], [4], [5], [6], [7]. These methods are typically based on the assumption that probabilistic data is available; however, this is not always true. Time-series data collected from participatory sensing is an important example where data processing is currently not widely applicable due to the lack of probability values. In an effort to rectify this situation, we focus on the problem of creating *probabilistic time series* from given (uncertain) time series collected by participatory sensing.

One of the most important challenges in creating probabilistic time series from time series is to deal with evolving probability distributions, since time series often exhibit highly irregular dependencies on time [6], [8]. For example, temperature changes dramatically around sunrise and sunset, but changes only slightly during the night. This implies that the probability distributions that are used as the basis for deriving probabilistic data also change over time, and thus must be computed dynamically. To this end, we identify and adopt a novel class of dynamical models from the time-series literature, which is known as GARCH (Generalized Auto Regressive Conditional Heteroskedasticity) model [9]. We show that the GARCH model can play an important role in efficiently and accurately creating probabilistic time series, by inferring dynamic probability distributions.

Another important challenge in creating probabilistic data from time series is how to deal with the sheer amount of sensors. A community of sensors may contain hundreds or thousands of nodes, which make it extremely difficult to handle if we deal with the data from each sensor alone. To make matters worse, the quality of the sensors are highly different. For example, a sensor may suffer from internal problems such as discharged batteries or external problems such as bad weather, which renders it to function improperly. As a result, the quality of the data collected from the sensors varies from sensor to sensor and from time to time. To circumvent these crucial problems, we propose a method to combine the probabilistic time series from numerous sensors based on their quality. By combining data from multiple sensors, we are able to achieve higher accuracies than using a single sensor alone. Our approach is based on an observation that there is an inter-dependence between the reliability of the sensors and their data. The quality of the data depends on the sensor that produces them while the reliability of sensors depends on the quality of the data they produce. As a result, we employ a trust model where the trustworthiness of the sensors and their data are measured concurrently and explicitly. More precisely, each sensor is associated with a trust score, which shows its accuracy. The data of the sensor is also assigned a trust score, which represents its probability of correctness. These trust scores provide us a meaningful way to combine probabilistic time series from multiple sensors.

Our contributions and the outline of this paper can be summarized as follows.

- Section II: We formally discuss the elements and the problem we want to solve. We also provide an overview of our approach to this problem. The approach requires realization of two components: computing likelihoods and aggregating probabilistic data. The former is responsible for generating probabilistic time series from time series while the latter combines probabilistic time series from multiple sensors.
- Section III: We adopt a novel approach to generate probabilistic time series from sensor data. This approach models a reading from each sensor by a probability distribution. We employ this approach for its ability to deal with evolving probability distributions.
- Section IV We propose an algorithm to effectively measure the trust scores of both the sensors and their data. The trust scores are computed based on the mutually reinforcing relationship between the sensors and the data they provide. Based on the trust scores of the sensors and their data, we discuss a method to combine the probabilistic time series from all sensors.
- Section V: We extensively evaluate our methods by performing experiments on both synthetic and real-world datasets. While the real-world datasets provide a pragmatic view, the synthetic dataset provides different settings where the real datasets can not cover.

The remaining sections are organized as follows. Section V demonstrates experimental results. Section VI summarizes related works, before Section VIII concludes the paper.

II. Model and Problem Statement

In this section, we first describe the elements of our problem. Secondly, we state the problem we want to solve formally. Then, we give an overview of our solution to the problem.

A. Model

We formally define the elements of our problem as follows. Let $\mathcal{D} = \{S_1, S_2, \cdots, S_n\}$ be a set of time series where a time series $S_i = \langle r_1^i, r_2^i, \cdots, r_m^i \rangle$ is a sequence of timestamped values. Each timestamped values $r_j^i \in S_i$ specifies a reading collected by sensor i at time j. Since sensors are not reliable, the raw data it provides at a timestamp may not be correct. The *expected true value* may resides within a range where each value in this range has a different probability to be the expected true value, hence, a distribution. We use a probability density function $p_j(R_j^i)$ to describe this distribution where R_j^i is a random variable associated with the reading r_j^i.

From the sensor data collected by participatory sensing, we aim to generate probabilistic sensor data which consists of various probabilistic time series. We denote $p\mathcal{D} = \{pS_1, pS_2, \cdots, pS_n\}$ as a set of *probabilistic time series* where a probabilistic time series $pS_i = \langle p_1(R_1^i), p_2(R_2^i), \cdots, p_m(R_m^i) \rangle$ constructed from a time series S_i is a sequence of probability density functions $p_j(R_j^i)$.

From the set of probabilistic time series $p\mathcal{D}$, we want to generate a combined probabilistic time series $\mathcal{G} = \langle p_1(G_1), p_2(G_2), \cdots, p_m(G_m) \rangle$. Each probability density function $p_i(G_i) \in \mathcal{G}$ is aggregated from the set of probability distributions provided by all sensors at timestamp t, which we denote as $\Omega_t = \{p_t(R_t^1), p_t(R_t^2), \cdots, p_t(R_t^n)\}$. We provide a summary of notations used in this paper in Table I.

TABLE I: Summary of Notations

Symbol	Description
S_i	A time series.
$\mathcal{D}$	A set of time series.
r_t^i	Raw (imprecise) value at time t of time series S_i.
R_t^i	Random variable associated with r_t^i.
$\hat{r}_t^i, \mathbb{E}(R_t^i)$	Expected true value at time t.
$p_t(R_t^i)$	Probability density function of R_t^i at time t.
$p\mathcal{D}$	A set of probabilistic time series.
pS_i	A probabilistic time series.
Ω_t	A set of probability density functions at time t.
$p_t(G_t)$	An aggregated probability density function at time t
$\mathcal{N}(\mu, \sigma^2)$	Normal (Gaussian) probability density function with mean μ and variance σ^2.
$\mathcal{G}$	The aggregated time series.

B. Problem statement

In this paper, we want to leverage the wisdom of the crowd by combining all the sensor data to one probabilistic time series $\mathcal{G} = \langle p_1(G_1), p_2(G_2), \cdots, p_m(G_m) \rangle$ where G_j is a random variable that represents all the data emitted by the sensors at timestamp j. Formally, our objective is defined in Problem 1.

Problem 1. ***Inference of probability distributions*** *Given a set of time series $\mathcal{D}$, the inference of probability distributions is the identification of the probabilistic time series $\mathcal{G} = \langle p_1(G_1), p_2(G_2), \cdots, p_m(G_m) \rangle$.*

Problem 1 is basically about generating probabilistic data from raw data and aggregating data from various sensors. We provide an overview of our approach in the following.

C. Approach overview

Figure 1 represents an overview of our framework. We start with the time series data collected by participatory sensing. From the time series data, we generate probabilistic time series data by means of the *Computing Likelihoods* component. As shown in Figure 1, a probabilistic time series differs from a time series in how we present the data at each timestamp. While an exact value is used in a time series, a probability distribution is used to approximate the expected true value at each timestamp in a probabilistic time series. The output of this component is a set of probabilistic time series, where each sensor is represented by a probabilistic time series.

In order to leverage the wisdom of the crowd and handle the huge amount of data from numerous sensors, we propose to aggregate the probabilistic time series. To this end, we combine all the probabilistic time series via the *Aggregate* component. Since the sensors are not reliable, in order to aggregate the probabilistic time series, we first need to measure the reliability of the sensors and their data. We achieve this by computing the trust scores of the sensors and their data. The higher the trust score of a sensor, the more trustworthy the sensor is. The *Trust Assessment* subcomponent in the *Aggregate* component is responsible for calculating the trust scores of the sensors while the *Trust-based aggregation* subcomponent takes the trust scores and aggregates the probabilistic time series.

Following this general structure, our framework requires the realization of the following components.

Computing Likelihoods. This component is responsible for generating probabilistic time series from the sensor data. It takes as input a set of time series from all the sensors and returns a set of probabilistic time series. We achieve this by adopting a dynamical model named GARCH. Although GARCH model has been used widely, its interpretation and application in this context is quite novel. For each time series it takes, the GARCH model returns a probabilistic time series, which contains a probability distribution at each timestamp. In order to compute this distribution, the GARCH model needs to measure two parameters: mean and variance. The detail of the computation and the GARCH model are discussed in Section III.

Aggregate. Given a set of probabilistic time series generated by the *Computing Likelihoods* component, the *Aggregate* component combines all probabilistic time series into one probabilistic time series. It consists of two subcomponents *Trust assessment* and *Trust-based aggregation*.

- *Trust assessment*: This subcomponent takes a set of probabilistic time series as input and assigns trust scores to each sensor and its data. It achieves this by concurrently measuring the trust score of the sensor and its data. A distribution has a high trust score if it is proposed by many reliable sensors. On the other hand, the trust score of a sensor is high if the data it provides also have high trust scores.
- *Trust-based aggregation*: Based on the trust scores, the Trust-based aggregation component is able to aggregate the probabilistic time series. Instead of treating the sensors and their data equally, we apply a weighted combination of the probabilistic time series based on the trust scores. Consequently, the sensors with high trust scores contribute more to the aggregated probabilistic time series and vice versa.

The *Aggregate* component will be described in detail in Section IV.

III. Computing likelihoods of sensor data

In this section, we describe the first component of our framework: the *Computing Likelihoods* component which is used to infer probabilistic time series from raw data. We adopt a novel method named GARCH (Generalized AutoRegressive Conditional Heteroskedasticity) model, which takes a time series as input and returns a probabilistic time series.

Recall that a probabilistic time series pS is a set of probability distributions $pS = \langle p_1(R_1), p_2(R_2), \cdots, p_m(R_m)\rangle$ where each probability density function $p_i(R_i)$ describes a distribution where the expected true value $\hat{r}_i$ resides. The GARCH method models $p_i(R_i)$ by a Gaussian probability density function $\mathcal{N}(\hat{r}_i, \hat{\sigma}^2)$. The function is specified by two parameters: an expected true value $\hat{r}_i$ and a variance $\hat{\sigma}$. We first discuss the method to predict the expected true value $\hat{r}_i$ using the AutoRegressive Moving Average (ARMA) model. Then, we descibe the GARCH model [9] to compute the variance $\hat{\sigma}$. Finally, we provide a unified algorithm to compute the probability density function $p_i(R_i)$ using both ARMA and GARCH model.

Predicting expected true values. The expected true values can be predicted using the *AutoRegressive Moving Average* (ARMA) model [9], which is commonly used for predicting expected values in time series [10]. Given a time series S, a reading at timestamp i can be modeled by the ARMA model as $r_i = \hat{r}_i + a_i$, where $\hat{r}_i$ is the expected true value and a_i follows a normal distribution $\mathcal{N}(0, \sigma_a^2)$. Then, the expected true value $\hat{r}_i$ at time i can be calculated from its past values as follows:

$$\hat{r}_i = \delta_0 + \sum_{j=1}^{p} \delta_j r_{i-j} + \sum_{j=1}^{q} \gamma_j a_{i-j}, \qquad (1)$$

where $p \geq 0$, $q \geq 0$ are respectively the autoregressive and moving average orders, $\delta_1, \ldots, \delta_p$ are autoregressive coefficients, $\gamma_1, \ldots, \gamma_q$ are moving average coefficients, δ_0 is a constant, and $i > max(p, q)$. Interested readers can consider [9] for a detailed discussion on the ARMA model.

Inferring variances. As the ARMA model models a reading r_i at time i as $r_i = \hat{r}_i + a_i$, we can then define the conditional variance given all the information available until time $i-1$ as

$$\sigma_i^2 = \mathbb{E}(a_i^2 | F_{i-1}), \qquad (2)$$

where $\mathbb{E}(a_i^2 | F_{i-1})$ is the variance of a_i. Then, based on the GARCH model, we can measure the variance as a linear function of a_i^2 as:

$$a_i = \sigma_i \epsilon_i, \quad \sigma_i^2 = \theta_0 + \sum_{j=1}^{h} \theta_j a_{i-j}^2 + \sum_{j=1}^{s} \lambda_j \sigma_{i-j}^2, \qquad (3)$$

where (h, s) specifies the model orders, ϵ_i is a sequence of independent and identically distributed (*i.i.d*) random variables, $\theta_0 > 0$, $\theta_j \geq 0$, $\lambda_j \geq 0$, $\sum_{j=1}^{max(h,s)} (\theta_j + \lambda_j) < 1$. Regarding the model orders, we set $h = 1, n = 1$ following common practice as it is very hard to specify the model order for higher order GARCH model[9].

We adopt the GARCH model to calculate the variances for its ability to deal with evolving probability distributions. More precisely, through the variable a_i, we are able to capture the high variance at timestamp i. Moreover, we can also model the changing trend in the time series where a high variance at timestamp i leads to a high variance at timestamp $i+1$.

Algorithm. We describe the detail of the approach to compute the probability density function in Algorithm 1. ARMA model is used to compute the expected true value $\hat{r}_t$ in Step 3

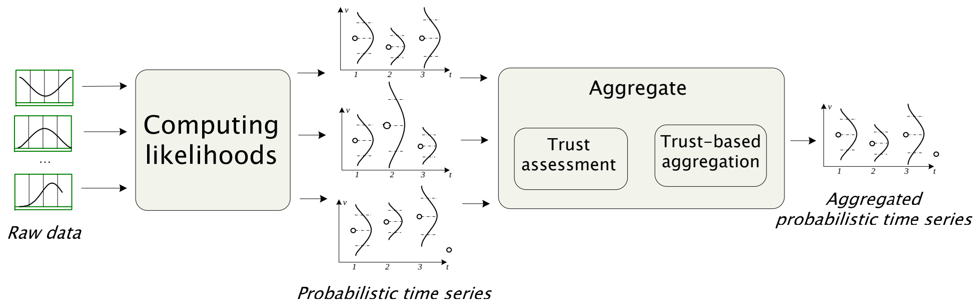


Fig. 1: Architecture of the framework.

while the GARCH model is used to calculate $\hat{\sigma}_t^2$ in Step 4. Regarding the complexity of the algorithms, the estimation steps (step 1 and 2) constitute the main part as its complexity is mainly affected by them. Their complexity is respectively $\mathcal{O}(m \cdot max(p,q))$ and $\mathcal{O}(m)$ for ARMA model and GARCH model.

Algorithm 1 Inferring $\hat{r}_t$ and $\hat{\sigma}_t^2$ using GARCH.

```
Input: A time series $S_i$, ARMA model parameters $(p,q)$ and scaling factor $\kappa$.
Output: Inferred $\hat{r}_t$, inferred variance $\hat{\sigma}_t^2$
1: Estimate an $ARMA(p,q)$ model on $S_i$ and obtain $a_i$.
2: Estimate a $GARCH(1,1)$ model using $a_i$'s
3: Infer $\hat{r}_t$ using $ARMA(p,q)$
4: Infer $\hat{\sigma}_t^2$ using $GARCH(1,1)$
5: return $\hat{r}_t$, $\hat{\sigma}_t^2$
```

IV. AGGREGATING MULTIPLE SENSOR DATA

In practice, a community of sensors contains thousands of sensors, which makes it extremely difficult to deal with if we take into account the time series from each sensor separately. As a result, we need to combine the time series emitted from the sensors. Moreover, by combing the data from the sensors, we can leverage the wisdom of the crowd to obtain a higher accuracy for the combined data. One naive approach is to combine the probabilistic time series from the sensors without considering the reliability of the sensors. However, as the quality of the sensors are not the same, this approach misses an important information. To this end, we propose a method to combine probabilistic time series based on the *trust scores*. The trust scores are used to measure the reliability of the sensors and the quality of their data.

The *Aggregate* component takes as input a set of probabilistic time series $p\mathcal{D} = \{pS_1, pS_2, \cdots, pS_n\}$, which are the results of the *Computing Likelihoods* component. It returns an aggregated time series $G = \langle p_1(G_1), p_2(G_2), \cdots, p_m(G_m)\rangle$ where $p_j(G_j)$ is the aggregated probability density function. The aggregated probability density function $p_j(G_j)$ at timestamp j is computed from the set of probability density functions $\Omega_j = \{p_j(R_j^1), p_j(R_j^2), \cdots, p_j(R_j^n)\}$ provided by the sensors at timestamp j. In the following, we first discuss our approach to compute the trust scores of the sensors and their data. Then, we explain how to aggregate the probabilistic time series based on the trust scores.

A. *Measuring trust score of sensor data*

Given a set of probability density functions of all sensors at timestamp j: $\Omega_j = \{p_j(R_j^1), p_j(R_j^2), \cdots, p_j(R_j^n)\}$, we need to estimate a trust score α_j^i for each probability density function $p_j(R_j^i) \in \Omega_j$. In order to compute the trust scores, we consider two factors:

- *Similarity of distributions*: distributions which are *similar* to each other should have higher trust scores. The reason is that similar distributions reinforce each other which makes them have a higher chance to be correct. For example, if 5 out of 8 distributions are similar, we tend to consider these 5 distributions correct based on majority rule. Although there may be minor difference between them, the differences may come from small errors in transmission, etc., which can be ignored.
- *Reliability of sensors*: we observe that a reliable sensor tends to provide correct information. As a result, a distribution which comes from a reliable sensor should have a higher trust score.

Numerically, we can estimate the trust score of distribution $p_t(R_t^i)$ by taking into account the above factors as follows:

$$\alpha_t^i = \frac{\sum_{j=1..n, j\neq i} \beta_j s_t^{i,j}}{\sum_{j=1..n} \beta_j} \tag{4}$$

where $\beta_j \in [0,1]$ is the trust score of sensor j and $s_t^{i,j} \in [0,1]$ is the similarity score of two probability density functions $p_t(R_t^i), p_t(R_t^j)$. The domain value of α_t^i is $[0,1]$ where the higher value of α_t^i, the higher chance that the distribution $p_t(R_t^i)$ can reflect exactly the true value at timestamp t. Note that $\alpha_t^i = 1$ means that all the distributions are the same ($\forall j \neq i, s_t^{i,j} = 1$), whereas $\alpha_t^i = 0$ means that all the trust scores of other sensors are zero ($\forall j \neq i, \beta_j = 0$).

Intuitively, Equation (4) models the relationship between distribution $p_t(R_t^i)$ and its neighbors (the distributions in Ω_t). If it is similar to a neighbor distribution which is provided by a reliable sensor, it should receive a high trust score. On the other hand, the trust score of distribution $p_t(R_t^i)$ should be low if the distribution is significantly different from the neighbors or the trust scores of the neighbor's sensors are low. For example, when the trust score of the neighbor's sensor is 0, although the two distributions are very similar $s_t^{i,j} \approx 1$, the neighbor distribution $p_t(R_t^j)$ does not contribute to the trust score of distribution $p_t(R_t^i)$.

In order to compute Equation (4), we need to calculate the similarity between two distributions and the trust scores of the sensors. In the following, we first discuss the method to measure the similarity between the distribution. Then, we describe our approach to compute the trust scores of the sensors.

Similarity of distributions. Regarding the first task, we employ the *Kullback-Leibler(KL) divergence*. The KL divergence between two probability density functions p and q is measured as follows:

$$KL(p \parallel q) = \int p(x) log(\frac{p(x)}{q(x)})$$

The divergence value between two distributions is non-negative but it is asymmetric. As a result, to use it as a measure, we employ the following symmetric variation and define it as the dissimilarity between two distributions:

$$dist(p,q) = KL(p \parallel q) + KL(q \parallel p)$$

However, as Equation 4 takes the similarity of the distributions as input, we need to calculate the similarity between two distribution from the KL divergence. To this end, consider a set of distributions at timestamp t $\Omega_t = \{p_t(R_t^1), p_t(R_t^2), \cdots, p_t(R_t^n)\}$ provided by the sensors, the similarity score between two distributions can be computed as follows:

$$s_t^{i,j} = 1 - \frac{dist(p_t(R_t^i), p_t(R_t^j))}{dist_{\Omega_t}} \tag{5}$$

where $dist_{\Omega_t}$ is the diameter of the distributions in Ω_t. The similarity score between the distributions takes a value in $[0, 1]$ where the higher the similarity score, the more similar between the distributions. Note that $s_t^{i,j} = 1$ means the two distributions are the same, where $s_t^{i,j} = 0$ only means the two distributions are the least similar pair in the set.

Estimating trust scores of sensors. We observe that a sensor that provides more correct data tends to be more reliable. Based on this observation, we can compute the trust scores of the sensors based on the trust scores of the data they provide. As a result, the trust score of a sensor can be calculated as follows:

$$\beta_j = \frac{\sum_{k=1}^{m} \alpha_k^j}{m} \tag{6}$$

Or informally we can say that the trust score of a sensor is measured by the average of the trust scores of all the distributions it provides. The domain of β_j is [0,1]. A high value of β_j represents the high reliability of sensor j. Note that $\beta_j = 1$ means that the sensor provides all distributions with trust scores equal to 1, whereas $\beta_j = 0$ means that the sensor does not provide any distribution with positive trust score.

Algorithm. We observe that there is an inter-dependence between Equation (4) and Equation (6). More specifically, in order to compute the trust score of a sensor β_j, we need to know the trust scores α_i of the distributions it provides. Nevertheless, estimating the trust score of a distribution α_i requires knowing the trust score of the sensors. There is a mutually reinforcing relationship between the sensor and the data it provides. As a result, we leverage the relationship between them via an iterative algorithm that maintains and updates the trust scores of the sensors and the distributions. We discuss the detail of the iterative algorithm in Algorithm 2.

Algorithm 2 Iterative Algorithm to Compute Trust Scores.

```
Input: A set of probabilistic time series $p\mathcal{D} = \{pS_1, pS_2, \cdots, pS_n\}$, a termination condition $\Delta$
Output: A set of trust scores $\alpha_t^i, \beta_j$.
 1: // Initialization
 2: $\beta_1^0 = 0.5$;...;$\beta_n^0 = 0.5$
 3: $q = 1$
 4: while $\Delta$ do
 5:     for $l = 1..m$ do
 6:         for $i = 1..n$ do
 7:             $\alpha_l^{i,q} = \frac{\sum_{j=1..n, j \neq i} \beta_j^{q-1} m_l^{i,j}}{\sum_{j=1..n} \beta_j^{q-1}}$
 8:     for $j = 1..n$ do
 9:         $\beta_j^q = \frac{\sum_{k=1}^{m} \alpha_k^{j,q}}{m}$
10:     $q = q + 1$
```

Algorithm 2 takes as input a set of probabilistic time series $p\mathcal{D} = \{pS_1, pS_2, \cdots, pS_n\}$ and returns a set of trust scores for the data and the sensors. We first initialize the trust scores of the sensors to 0.5 in Line 1. The algorithm iterates between two phases: estimating the trust scores of the sensors and the data. In order to estimate the trust score $\alpha_l^{i,q}$ of the data at iteration q, we use the trust scores of the sensors β_j^{q-1} which are estimated in the previous iteration (Line 5-7). Then, in the second phase, we use the trust scores of the data $\alpha_l^{i,q}$ to estimate the trust scores of the sensors β_j^q in this iteration (Line 8-9). We continue the iteration until the termination condition Δ is satisfied. Note that we can reverse the computation by estimating the trust scores of the sensors β_j^q before estimating the trust scores of the data α_i^q. However, this requires an initialization of the trust scores for all the distributions, which is not feasible if the time series is long. Regarding the termination condition Δ, in most of the cases, we want to stop the algorithm when the changes of the trust scores between two iterations are negligible.

B. Trust-based aggregation

After the previous step, we have acquired a set of trust scores for the distributions and the sensors. Given the distributions at a timestamp t, we need to generate an aggregated distribution that can both represent the distributions and the reliability of the sensors. To this end, we first need to model the relationship between the the aggregated distribution and the distributions at timestamp t. Based on this relationship and the trust scores, we can generate the aggregated distribution effectively.

Relationship between aggregated and constituent random variables. Since the aggregated distribution at timestamp t is generated from the distributions at the same timestamp, the random variable G_t which models the aggregated distribution should be computed from the random variables R_t^i of the distributions. Another observation is that the contribution of the random variables R_t^i to G_t is different. Intuitively, a distribution which has a higher trust score contributes more to the aggregated distribution. It comes from the fact that the distribution with the higher trust score can express the true value at timestamp t better.

Based on these observations, we can model the relationship between the aggregated and constituent random variables as a weighted sum as follows:

$$G_t = \frac{\sum_{i=1}^{n} \alpha_i R_t^i}{\sum_{i=1}^{n} \alpha_i} \quad (7)$$

Compute aggregated probability density functions. Given the random variables of the distributions at timestamp t, we can compute the aggregated random variable at the same timestamp based on Equation 7. Let $M_{R^i}(.)$ be the *moment-generating function*[11] of the random variable R^i. Since G is the linear combination of independent random variables $R^1, R^2, ..., R^n$, its moment-generating function is as follows:

$$M_G(x) = \prod_{i=1}^{n} M_{R^i}(\alpha_i x)$$

However, recall that in our GARCH model, the random variable R^i at timestamp t models a normal distribution $\mathcal{N}(\hat{r}, \hat{\sigma}^2)$. Since R^i is a normal distribution $\mathcal{N}(\hat{r}_i, \hat{\sigma}_i^2)$, its moment-generation function M_{R^i} is as follows:

$$M_{R^i}(x) = \exp(\hat{r}_i x + \frac{\hat{\sigma}_i^2 x^2}{2})$$

Therefore, we have

$$M_G(x) = \prod_{i=1}^{n} \exp(\hat{r}_i \alpha_i x + \frac{(\hat{\sigma}_i^2)^2 \alpha_i^2 x^2}{2})$$
$$= \exp(x(\sum_{i=1}^{n} \alpha_i \hat{r}_i) + \frac{x^2}{2}(\sum_{i=1}^{n} \alpha_i \hat{\sigma}_i^2))$$

However, this equation has the same structure as the moment-generating function for a normal random variable $\mathcal{N}(\sum_{i=1}^{n} \alpha_i \hat{r}_i, \sum_{i=1}^{n} \alpha_i \hat{\sigma}_i^2)$. From the uniqueness property of moment generating functions, G must be a normal random variable which follows the following normal distribution $\mathcal{N}(\sum_{i=1}^{n} \alpha_i \hat{r}_i, \sum_{i=1}^{n} \alpha_i \hat{\sigma}_i^2))$.

As a result, from the distributions generated in Section III and the trust scores generated in Section IV-A, we are able to generate an aggregated distribution $p_t(G_t) = \mathcal{N}(\sum_{i=1}^{n} \alpha_i \hat{r}_{i,t}, \sum_{i=1}^{n} \alpha_i \hat{\sigma}_{i,t}^2))$ at timestamp t. Applying this approach for other timestamps, we are able to generate an aggregated probabilistic time series $G = \langle p_1(G_1), p_2(G_2), \cdots, p_t(G_t) \rangle$ from the data collected from the sensors.

V. Experimental Evaluation

This section presents a comprehensive experimental evaluation of the proposed method using both real-world and synthetic datasets. The results show that the presented approach supports generating probabilistic data effectively and efficiently. We proceed as follows: we first discuss the datasets we use for the experiments. Then, we report the results of computing likelihoods, computing trust scores and aggregating probabilistic data.

A. Datasets

In our experiments, we use two types of data: real-world data and synthetic data. While the real-world data shows us a pragmatic view of the real settings, the synthetic data provides different settings which the real-world dataset cannot cover.

Real-world Data. We relied on two real-world datasets covering different domains where participatory sensing can be applied. (1) *Campus*: this dataset contains temperature readings collected from a real sensor network deployed on the EPFL university campus. The dataset is built over twenty five days and includes about eighteen thousand samples. It is referred in this paper under the name *campus-data*. (2) *Moving Object*: the dataset contains about ten thousand GPS readings collected over five and a half hours from 192 cars in Copenhagen, Denmark. Each log entry contains the time and both the x-y coordinates. In this dataset, which we refer as *car-data*, only the x-coordinate is used in the experiments.

TABLE II: Summary of Datasets

	campus-data	*car-data*
Monitored parameter	Temperature	GPS Position
Number of data values	18031	10473
Sensor accuracy	± 0.3 deg. C	± 10 meters
Sampling interval	2 minutes	1-2 seconds

An important observation regarding the real-world datasets is that the *car-data* dataset is smoother than the *campus-data* one. This is expected as cars always follow a predefined path of roads and highways, which makes the changes in x-y coordinates smooth. We summarize the important properties of the real dataset in Table II.

Synthetic Data. In order to evaluate our proposed method in various scenarios, we generate a synthetic dataset to be able to control the parameters. At each timestamp, we first generate a true distribution which we assume to best model the expected true value for this timestamp. Then, at the same timestamp, we construct a normal distribution for each sensor which is assumed to present the readings collected from the sensor. The normal distribution is constructed from the true distribution by adding a small difference to the mean of the true distribution. By applying this method to all the timestamps, we are able to generate the synthetic dataset.

B. Evaluations on computing likelihoods

In this experiment, we want to analyze the running time of the Computing Likelihoods component (i.e. the GARCH model). In order to fairly measure the performance of the GARCH model, we use the two real-world datasets (*campus-data* and *car-data*) and vary the length of the time series from 30 to 180.

The experimental results are shown in Table III. We can observe that the running time of the GARCH model is low which makes it suitable for online and realtime applications. Another interesting observation is that the running time increases as we increase the length of the time series. However, the increase in running time is slower than the increase in time series length. For example, when the length of the time series increases 6

TABLE III: Running time of the GARCH method ($\log_2(s)$)

Time series length	campus-data	car-data
30	0.1314	0.1205
60	0.1543	0.1419
90	0.182	0.1653
120	0.2092	0.1874
150	0.2379	0.2104
180	0.2634	0.2333

times from 30 to 180, the running time of the GARCH model only doubles. In summary, the results conclude the fact that the computation of our GARCH model is efficient for real-world datasets.

C. *Evaluations on computing trust*

In the following experiments, we want to analyze the Trust assessment component in different aspects, including *computation time*, *convergence*, *effects of outliers*, and *effects of distance*.

Computation time. In practice, data from the sensors are used in various applications which require immediate response. As a result, the computation time to calculate the trust scores is critical to the performance of these applications. In this experiment, we want to analyze the running time of our algorithm with different settings. More specifically, we vary the number of sensors from 10 to 30 and the length of the time series from 100 to 400. The experimental result is illustrated in Table IV.

A significant observation is that the running time increases linearly with the increase in the length of the time series. For instance, when the number of timestamps triples from 100 to 300, the computation time also increases three times despite of the change in the number of sensors. On the other hand, when the number of sensors increases, the running time does not increase linearly. The reason is that we need to measure the similarity between every pair of distributions provided by the sensors. As the number of sensors increases, the number of pairs increases significantly.

Effect of outliers. In this experiment, we want to analyze the accuracy of the proposed algorithm in computing the trust scores. To this end, we generate some outliers (sensors which data are completely different from the rest) among the sensors. In reality, some sensors may produce data that are completely different from other sensors. The reason is that their communication channels are blocked by buildings, walls or their processing units have severe problems. In these cases, applications or methods that use the data provided by these spammer sensors may produce incorrect results. By increasing the percentage of outliers in the network of sensors from 10 to 50%, we want to verify our hypothesis that the outliers receive low trust scores. The experimental results is shown in Fig. 2. The X-axis and the Y-axis represents the percentage of outliers and the trust score, respectively.

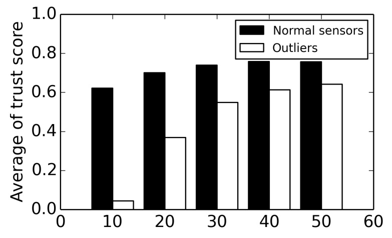


Fig. 2: Accuracy of the algorithm

A key observation is that the outliers and normal sensors can be clearly separated. At each value of the percentage of outliers, the normal sensors which always have high trust scores while the outliers always have lower trust scores. This distinction shows that our algorithm is able to distinguish between normal sensors and outliers. Another interesting finding is that the trust scores of normal sensors are still higher than the outliers even the percentage of outliers is high (40 or 50%).

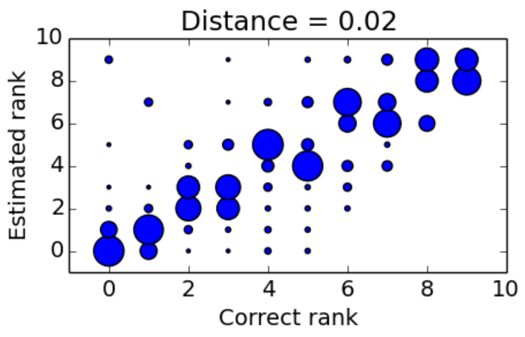
Effects of distance between distributions.. In the previous experiment, we have demonstrated the accuracy of the algorithm when calculating the trust scores of the sensors. In this experiment, we want to analyze the accuracy of the algorithms regarding the trust scores of the data i.e., the distributions. Recall that in order to construct the synthetic dataset, we first generate a true distribution at each timestamp. Based on the true distribution, we generate the distributions of the sensors by adding small differences to the mean of the true distribution. Now after generating the synthetic dataset, we rank the distributions at a timestamp by how close it is to the true distribution. The closer a distribution to the true distribution, the higher rank it receives. After running Algorithm 2, we calculate the rank of the distribution again where the distribution with the higher trust score has the higher rank. The experimental result is shown in Fig. 3, in which we vary the distance between the distributions (0.02 and 0.2). The X-axis is the correct ranking (w.r.t true distribution), while the Y-axis is the estimated ranking (w.r.t trust score).

Each point in the two plots of Fig. 3 represents a distribution located by the ranking of the distribution before and after running the algorithm. As a result, there may be many distributions that could have the same true rank and estimated rank. We illustrate this factor by the size of the point: the larger the point is, the more distributions that have the same true rank and estimated rank at that point.

Intuitively, the closer the point to the diagonal, the better since it shows that the true rank of a distribution measured before and the estimated rank measured after running the algorithm are nearly the same. This means the algorithm is able to assess the trust scores of the distributions correctly. A key observation from Fig. 3 is that the number of points located near the diagonal is significantly high. This shows that our

TABLE IV: Running time of Algorithm 2 (ms)

#Sensors	#Timestamps			
	100	200	300	400
10	1848	3665	5499	7395
20	21256	23708	35613	46953
30	61066	123155	183260	245790



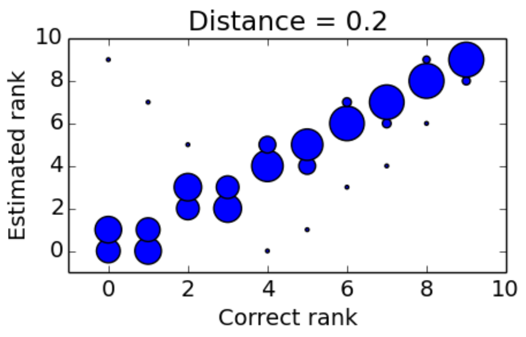


Fig. 3: Effect of distance between distributions on the accuracy of computing trust

algorithm is able to estimate the ranks of the distributions correctly, i.e., the trust scores of the distributions are also correct. We also observe that by increasing the distance between the distributions, the number of points located near the diagonal increase. Therefore, it shows that the accuracy of our algorithm increases if the differences between the distributions are high.

D. Evaluations on aggregating distributions

We now analyze the performance of our aggregating component via the following experiments.

Effects of heterogeneity level. As mentioned in previous section, quality of the sensors are different from one sensor to another. As a result, the data generated by the sensors are also different i.e., the heterogeneity level among the sensors are high. In this experiment, we want to analyze the effect of heterogeneity level to the performance of our proposed method. More specifically, we vary the difference between the means of the distributions and the mean of the true distribution from 1 to 20. The experimental result is shown in Fig. 4. The X-axis is the relative difference between the two means, whereas the Y-axis is the distance against the true distribution.

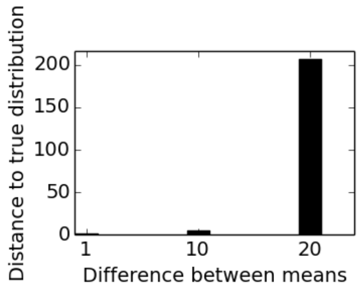


Fig. 4: Distance to true distribution w.r.t percentage of change in mean

An interesting observation is that the distance to the true distribution increases as the difference increases. This can be explained as follows: since we increase the difference between the means of the distributions of the sensors and the mean of the true distribution, the distributions of the sensors become deviating from the true distribution. As a result, the more different between the distributions and the true distribution, the more different between the aggregated distribution and the true distribution. Therefore, the distance between the true distribution and the aggregated distribution increases as the difference increases.

Effects of outliers. In this experiment, we want to find out the effects of the outlier sensors to the performance of our Aggregating component. We measure the accuracy of our approach by comparing the distance of the aggregated distributions to the true distributions. The experimental result is illustrated in Fig. 5. The X-axis is the percentage of outliers among all sensors, while the Y-axis is the distance against the true distributions.

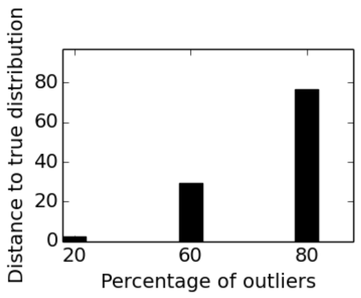


Fig. 5: Distance to true distribution w.r.t percentage of outliers

An interesting observation is that the distance against the true distribution increases as the number of outliers increases. This phenomenon is expected and can be explained as follows: as the number of outliers increases, they are able to pollute the data provided by correct sensors. However, as we expect correct sensors to provide distributions that are similar to the true distribution, the data provided by outliers will make the aggregated distribution different from the true distribution. Therefore, the more outliers, the more different between the aggregated distribution and the true distribution.

VI. Related Work

There is a large body of work about data stream (e.g. time series, sensor data), such as data compression [12], indexing [13], and pattern matching [14]. Whereas, our work studies about estimating the true value of data provided by sensors. In the following, we review the work in probabilistic databases, probabilistic data stream, and participatory sensing that is close to our research.

Probabilistic Databases. Recently, there has been a plethora of research on probabilistic databases including concepts and foundations [15], [16], [17], query processing [4], [18], [19], [20], and indexing schemes [21], [22]. The main reason for this is the considerable potential benefit of applying probabilistic databases to other domains. For example, an application of probabilistic database to the named entity resolution problem [23] has speed up the recognition process of the New York Times article corpus significantly. However, these applications require that probability values associated with the data are available, which is not always correct. One approach to solve this problem is proposed in [24] where the authors construct probabilistic databases by inferring probability distributions and building views containing probability values. While the main goal of [24] is to deal with indexing techniques and query processing on probabilistic databases for a single sensor, the main goal of this paper is to develop a probabilistic model for multiple sensors in participatory sensing.

Probabilistic Data Streams. Another domain related to probabilistic databases is *probabilistic data streams*[22], [25], [26], which has received a significant attention recently. A systematic approach to generate probabilistic data streams is discussed in [8] by Tran *et al.*. However, the framework in [8] only focuses on RFID data instead of generic data streams. In [7], Ré *et al.* introduce a query processing framework for probabilistic data streams. Then, an access method to improve the efficiency of query processing on probabilistic data stream is proposed in [22]. Another approach to query processing for probabilistic data streams is proposed by Cormode and Garofalakis in [6]. This approach employs hash-based sketch synopsis structure to process aggregate queries for probabilistic data stream. While the work in [8] also focuses on creating probabilistic data streams, our approach is different as we take into account the quality of the sensors in the process.

Participatory sensing. In recent years, there has been an increased interest on participatory sensing due to the increasing number of personal sensors such as smartphones, GPS devices, and cameras. Participatory sensing campaigns often recruit participants to provide measurements on a particular region or a population. The examples of these campaigns include monitoring traffic [27], measuring the level of pollution [28], or documenting quality of roads[29]. As the quality of the sensors are unknown beforehand, many methods to assess the quality of the sensors and find the correct data (truth finding) have been proposed. These truth finding methods aim to find the correct data from the readings provided by the sensors. An approach to tackle truth finding in participatory sensing is proposed in [30] where the authors employ the Expectation Maximization algorithm to estimate the correct data. Although our proposed approach also estimates the quality of the sensors, our approach is different from [30] as we focus on generating probabilistic databases instead of finding the correct values from the data. It is noteworthy that participatory sensing is quite similar to crowdsourcing, in which data are provided by multiple workers. Each worker also has a "trust" score that reflects the quality of his answers. In this sense, we could apply answer aggregation techniques [31] from the crowdsourcing setting to our setting. However, we should be careful about the main difference of input data between the two settings. That is, data provided by workers are multiple-choice answers; whereas, data provided by sensors are real values.

VII. Applications

In this section, we tentatively provide a list of applications that can be built on top of our model.

Guidance for data repair. Poor data quality cannot be avoided when collecting data from sensors. In practice, we often employ human knowledge from users to improve the quality of collected data. Since the amount of dirty data might be far more than one can expect a user to handle, techniques for minimizing user effort become necessary. Intuitively, to minimize user effort, we should suggest the most uncertain data to user. To quantify the uncertainty of data, one can apply the probabilistic model proposed in this work.

Adaptive data acquisition in resilience systems. The reliability of participatory sensing is subject to different factors that impact the availability and accuracy of the results, the first one being the mobility of the sensing nodes. The high energy consumption of sensors compared to the actual battery of the device may also push the users to set more conservative policies, not submitting data. And finally the actual internet connection of the device may suffer from loss of network connectivity.

To address these issues, we need to implement a resilience system for potential improvements of data quality. A core feature of such system is detecting the above unexpected situations. One can leverage our probabilistic model to implement this feature. For example, if a sensor data has probability distributions of consecutive timestamps that have high variances, it means that it might contain imprecise readings. As a consequence, it alerts us that the sensor that provides data could be in unexpected situations.

VIII. Conclusions and Future Work

In this paper, we build a systematic model to manage uncertain data from participatory sensors. We define the notion of probability distribution for a time series as the backbone of our approach. Most importantly, our probabilistic model allows the capture of the dynamic and uncertain nature of participatory sensing data. Our approach involves two steps. In the first step, we convert raw data from single sensor into probabilistic data by using GARCH methods. In the second step, we unify probabilistic data from multiple sensors by using trust-based aggregation. Finally, we presented a comprehensive experimental evaluation of each of the two steps, indicating that the approach is applicable for real-world datasets and allows for effective and efficient management of participatory sensing data.

Our work opens up some potential research directions. While this paper focuses on combining probabilistic data from multiple sensors, our techniques, especially trust-based aggregation method, can be applied for prediction tasks. For example, one would like to predict the future outcomes based on historical data. There are different prediction methods, each of which produce a different probability distribution. To improve the prediction performance, we can combine these distributions via the proposed algorithm.

Acknowledgment

The research has received funding from the EU-FP7 EINS project (grant number 288021), the EU-FP7 PlanetData project (grant number 257641), and the OpenSense project.

References

[1] R. Cheng, D. V. Kalashnikov, and S. Prabhakar, "Evaluating probabilistic queries over imprecise data," in *SIGMOD*, 2003, pp. 551–562.

[2] M. Hua, J. Pei, W. Zhang, and X. Lin, "Ranking queries on uncertain data: a probabilistic threshold approach," in *SIGMOD*, 2008, pp. 673–686.

[3] N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," *JVLDB*, pp. 523–544, 2007.

[4] D. Olteanu, J. Huang, and C. Koch, "Sprout: Lazy vs. eager query plans for tuple-independent probabilistic databases," in *ICDE*, 2009, pp. 640–651.

[5] Y. Tao, R. Cheng, X. Xiao, W. K. Ngai, B. Kao, and S. Prabhakar, "Indexing multi-dimensional uncertain data with arbitrary probability density functions," in *VLDB*, 2005, pp. 922–933.

[6] G. Cormode and M. Garofalakis, "Sketching probabilistic data streams," in *SIGMOD*, 2007, pp. 281–292.

[7] C. Ré, J. Letchner, M. Balazinksa, and D. Suciu, "Event queries on correlated probabilistic streams," in *SIGMOD*, 2008, pp. 715–728.

[8] T. Tran, C. Sutton, R. Cocci, Y. Nie, Y. Diao, and P. Shenoy, "Probabilistic inference over rfid streams in mobile environments," in *ICDE*, 2009, pp. 1096–1107.

[9] R. H. Shumway and D. S. Stoffer, *Time series analysis and its applications*. Springer-Verlag, 2000.

[10] D. Tulone and S. Madden, "Paq: Time series forecasting for approximate query answering in sensor networks," in *EWSN*, 2006, pp. 21–37.

[11] C. M. Grinstead and J. L. Snell, *Introduction to probability*. American Mathematical Soc., 1998.

[12] N. Q. V. Hung, H. Jeung, and K. Aberer, "An evaluation of model-based approaches to sensor data compression," *TKDE*, pp. 2434–2447, 2013.

[13] N. Q. V. Hung and D. T. Anh, "An improvement of paa for dimensionality reduction in large time series databases," in *PRICAI*, 2008, pp. 698–707.

[14] ——, "Combining sax and piecewise linear approximation to improve similarity search on financial time series," in *ISITC*, 2007, pp. 58–62.

[15] L. V. Lakshmanan, N. Leone, R. Ross, and V. S. Subrahmanian, "Probview: a flexible probabilistic database system," *TODS*, pp. 419–469, 1997.

[16] N. Dalvi and D. Suciu, "Management of probabilistic data: foundations and challenges," in *PODS*, 2007, pp. 1–12.

[17] A. Jha and D. Suciu, "Probabilistic databases with markoviews," in *VLDB*, 2012, pp. 1160–1171.

[18] C. Mayfield, J. Neville, and S. Prabhakar, "Eracer: a database approach for statistical inference and data cleaning," in *SIGMOD*, 2010, pp. 75–86.

[19] R. Akbarinia, P. Valduriez, and G. Verger, "Efficient evaluation of sum queries over probabilistic data," *TKDE*, pp. 764–775, 2013.

[20] M. Dylla, I. Miliaraki, and M. Theobald, "Top-k query processing in probabilistic databases with non-materialized views," in *ICDE*, 2013, pp. 122–133.

[21] B. Kanagal and A. Deshpande, "Indexing correlated probabilistic databases," in *SIGMOD*, 2009, pp. 455–468.

[22] J. Letchner, C. Re, M. Balazinska, and M. Philipose, "Access methods for markovian streams," in *ICDE*, 2009, pp. 246–257.

[23] M. Wick, A. McCallum, and G. Miklau, "Scalable probabilistic databases with factor graphs and mcmc," in *VLDB*, 2010, pp. 794–804.

[24] S. Sathe, H. Jeung, and K. Aberer, "Creating probabilistic databases from imprecise time-series data," in *ICDE*, 2011, pp. 327–338.

[25] T. Chen, L. Chen, M. T. Ozsu, and N. Xiao, "Optimizing multi-top-k queries over uncertain data streams," *TKDE*, pp. 1814–1829, 2013.

[26] R. Cheng, T. Emrich, H.-P. Kriegel, N. Mamoulis, M. Renz, G. Trajcevski, and A. Zufle, "Managing uncertainty in spatial and spatio-temporal data," in *ICDE*, 2014, pp. 1302–1305.

[27] P. Zhou, Y. Zheng, and M. Li, "How long to wait?: Predicting bus arrival time with mobile phone based participatory sensing," in *MobiSys*, 2012, pp. 379–392.

[28] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda, "Peir, the personal environmental impact report, as a platform for participatory sensing systems research," in *MobiSys*, 2009, pp. 55–68.

[29] S. Reddy, K. Shilton, G. Denisov, C. Cenizal, D. Estrin, and M. Srivastava, "Biketastic: sensing and mapping for better biking," in *CHI*, 2010, pp. 1817–1820.

[30] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher, "On truth discovery in social sensing: A maximum likelihood estimation approach," in *IPSN*, 2012, pp. 233–244.

[31] N. Q. V. Hung, N. T. Tam, L. N. Tran, and K. Aberer, "An evaluation of aggregation techniques in crowdsourcing," in *WISE*, 2013, pp. 1–15.