Privacy-Preserving Reconstruction of Multidimensional Data Maps in Vehicular Participatory Sensing

Nam Pham[1], Raghu K. Ganti[1], Yusuf S. Uddin[1], Suman Nath[2] and Tarek Abdelzaher[1]

[1] University of Illinois at Urbana-Champaign
[2] Microsoft Research
{nampham2, rganti2, mduddin2, zaher}@illinois.edu
sumann@microsoft.com

Abstract. The proliferation of sensors in devices of frequent use, such as mobile phones, offers unprecedented opportunities for forming self-selected communities around shared sensory data pools that enable community specific applications of mutual interest. Such applications have recently been termed *participatory sensing*. An important category of participatory sensing applications is one that construct maps of different phenomena (e.g., traffic speed, pollution) using vehicular participatory sensing. An example is sharing data from GPS-enabled cell-phones to map traffic or noise patterns. Concerns with data privacy are a key impediment to the proliferation of such applications. This paper presents theoretical foundations, a system implementation, and an experimental evaluation of a perturbation-based mechanism for ensuring privacy of location-tagged participatory sensing data while allowing correct reconstruction of community statistics of interest (computed from shared perturbed data). The system is applied to construct accurate traffic speed maps in a small campus town from shared GPS data of participating vehicles, where the individual vehicles are allowed to "lie" about their actual location and speed at all times. An extensive evaluation demonstrates the efficacy of the approach in concealing multi-dimensional, correlated, time-series data while allowing for accurate reconstruction of spatial statistics.

1 Introduction

An emerging category of applications focus on collecting and sharing sensor data for the purpose of characterizing aggregate real-world properties, such as computing community-wide statistics or mapping physical phenomena of common interest. These applications are termed *participatory* sensing applications [1]. Examples of these applications include vehicular sensor networks for collecting and sharing traffic data [2], bicycle networks to collect and share bikers' paths [3], and cell phone based buddy networks to collect and share location and activity information [4]. An important category of participatory sensing applications is one where users share location-tagged data to construct maps of different phenomena (e.g., traffic speed, pothole, pollution).

One main problem in participatory sensing applications that share location-tagged data is privacy. For example, a community of environmentalists might want to collectively measure pollution on city streets and share that information to construct city-scale pollution maps. Since such data are location-tagged, a key question is to enable correct geographic mapping without revealing private location information of individuals collecting the location-sensitive data. The problem becomes non-trivial in the absence of a shared trusted entity that can be used to sanitize the data. Moreover, since the data itself, such as GPS traces, may reveal user identity, anonymity is not the answer to the privacy problem.

To address the above challenge, in this paper, we solve the privacy problem via data perturbation. Perturbing data on the client-side prior to sharing empowers clients by giving them the freedom to "lie" about both their data and the context (such as location) where it was collected. Clients share their perturbed data with an entity we call the *aggregation server*. It is responsible for computing the aggregate statistics of interest. Clients trust the server with computing the statistics but do not want to reveal their private data to it for privacy reasons. When receiving perturbed data, in addition to computing the community statistics, the server may try to guess the original individual user data, which we call a *privacy attack* . This paper designs perturbation algorithms that protect against privacy attacks, while ensuring accurate reconstruction of community statistics. The contribution lies in solving the above problem for the case of multidimensional correlated time-series data (such as correlated sensor data streams).

From an algorithmic perspective, the fundamental limitation of previous approaches is that they do not consider privacy-preserving perturbation and reconstruction when each user shares *multiple correlated* private data streams. For example, when collecting speed at different locations to build a city speed map, both speed and location are private since a client might not want to admit, say, to speeding, and might not want their location to be tracked.

We provide a solution to the general problem of ensuring privacy for multi-stream data of individuals while allowing community statistics to be reconstructed accurately. We develop a correlated noise model that can be utilized for perturbing location-tagged data in a way that protects both data and location privacy. We evaluate the approach using a traffic monitoring application implemented using an existing architecture called PoolView [5]. The application follows a client-server model. The client-side software collects data from the client's GPS device, perturbs the data and shares those with an aggregation server. The aggregation server then estimates useful community statistics from perturbed data and makes those statistics available for community access. Empirical measurements show that the approach results in accurate reconstruction of speed maps from perturbed data while preventing the reconstruction of individual client data and location information.

The rest of this paper is organized as follows. We first develop the reconstruction algorithm of the joint probability distribution in Section 2. Privacy properties are discussed in Section 3. Section 4 and Section 5 describe simulation-based evaluation and deployment-based evaluation, respectively. Finally, Section 6 concludes the paper.

2 Joint Probability Density Function Reconstruction

The main contribution of this paper lies in the algorithm to accurately reconstruct the community joint density given the perturbed multidimensional stream data and the noise density information. Any statistical question about the community can be answered using the reconstructed joint density. There have been many efforts on the community distribution reconstruction. Agrawal et al. [6] proposed a Bayesian-based reconstruction of the probability distribution. In [7], the authors use the Expectation Maximization (EM) algorithm to estimate one-dimensional distribution from data perturbed with Gaussian noise. In our previous work [5], we employed the Tikhonov-Miller deconvolution technique to estimate the community distribution. However, all of these algorithms are developed to reconstruct a one-dimensional distribution. Hence, they do not scale to the problem of multidimensional distribution reconstruction. In this section, we present an iterative algorithm to estimate the discretized joint distribution of multidimensional data streams.

Let the number of data streams that each user wants to share be M. The shared data from each user are assumed to be drawn from a multivariate random variable $X = (X_1, X_2, \ldots, X_M)$, thus each data point is a length M vector. The reconstruction algorithm does not distinguish which data points are from which user. Therefore, we can define the set of all data points from all users as $\bar{X} = \{x_1, x_2, \ldots, x_n\}$ where x_i is a length M data point, and n is the total number of data points from all users.

Each data point is perturbed by adding an M-dimensional noise data point generated from a known joint distribution $f_N(N_1, N_2, \ldots, N_M)$ which is known to all participating users (or rather to their client-side software). An aggregation server receives the set of n perturbed data points from all users denoted as $\bar{Y} = \{y_1, y_2, \ldots, y_n\}$. We want to estimate the joint distribution of X which is $f_X(X_1, X_2, \ldots, X_M)$ given the shared data $\bar{Y}$ and the knowledge of the noise distribution f_N.

Let us denote the sample space of X_i as Ω_i. Thus, the sample space of X is $\Omega = \Omega_1 \times \Omega_2 \times \ldots \times \Omega_M$. In order to reconstruct the density of X, we first discretize the the sample space Ω. The sample space of X_i is partitioned into K_i bins (may not be uniform) denoted as $\{\Omega_i^1, \Omega_i^2, \ldots, \Omega_i^{K_i}\}$. Thus Ω containes $K = K_1 \times K_2 \times \ldots \times K_M$ M-dimensional bins in which the value of the density function is constant. The more the number of bins, the better the discrete density approximates the continuous density. To simplify the notation, the following symbols are introduced:

- ω_I: the I^{th} bin of Ω, thus $\Omega = \cup_{\omega_I} \omega_I$.

- $\Theta = \{\theta_1, \theta_2, \dots, \theta_K\}$: where $\theta_i = f_X(X)$ with $X \in \omega_I$, is the set of all density parameters to be estimated.
- m_{ω_I}: the volume of ω_I, a proper discrete density parameters Θ should satisfy

$$\sum_{\omega_I} \theta_I m_{\omega_I} = 1 \tag{1}$$

To estimate Θ, our approach is to employ the maximum likelihood framework. We need to find the density function parameters which maximize the log likelihood of the data $\bar{X}$ given the observations $\bar{Y}$

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmax}} \log f_{X;\Theta}(\bar{X}|\bar{Y}) \tag{2}$$

The notation $f_{X;\Theta}$ means that the likelihood of X is computed using the discrete distribution Θ. Unfortunately, the likelihood can not be computed directly at the aggregation server because only $\bar{Y}$ is known while $\bar{X}$ is missing. A common procedure to solve the maximum likelihood estimation with incomplete information is the EM algorithm [8]. To use the EM algorithm, the following auxiliary function $Q(\Theta|\hat{\Theta}^k)$ is defined:

$$Q(\Theta|\hat{\Theta}^k) = E_{X|Y}\left[\log f_{X;\Theta}(\bar{X})|\bar{Y}, \hat{\Theta}^k\right] \tag{3}$$

The auxiliary function Q is actually the expectation of the likelihood in (2) with respect to X using the density of X computed from the previous step which is $\hat{\Theta}^k$. The EM algorithm consists of two steps:

- E-step : Given the density computed from the k^{th} step, compute the value of $Q(\Theta|\hat{\Theta}^k)$
- M-step : Compute $\hat{\Theta}^{k+1} = \operatorname{argmax}_{\Theta} Q(\Theta, \hat{\Theta}^k)$

Next, we will derive a closed form expression for Q, the optimal solution which maximizes the likelihood function and analyze the convergence of the algorithm.

Theorem 1. *(E-step) The value of $Q(\Theta|\hat{\Theta}^k)$ is given by:*

$$Q(\Theta|\hat{\Theta}^k) = \sum_{\omega_I} \hat{\theta}^k_{\omega_I} \log(\theta_{\omega_I}) \phi^k_{\omega_I} \tag{4}$$

Where

$$\phi^k_{\omega_I} = \frac{1}{N} \sum_{j=1}^{N} \frac{f_N(y_j - \omega_I)}{f^k_{Y;\hat{\Theta}^k}(y_j)} \tag{5}$$

$$f_{Y;\hat{\Theta}^k}(y_j) = \sum_{\omega_I} f_N(y_j - \omega_I)\hat{\theta}^k_{\omega_I} \tag{6}$$

$$f_N(y_j - \omega_I) = \int_{\omega_I} f_N(y_j - \gamma) d\gamma \tag{7}$$

Proof. See Appendix 7.1.

Theorem 2. *(M-step) The value of $\hat{\Theta}^{k+1}$ maximizing the auxiliary function $Q(\Theta|\hat{\Theta}^k)$ is given by*

$$\hat{\theta}_{\omega_I}^{k+1} = \frac{\phi_{\omega_I}^k}{m_{\omega_I}} \hat{\theta}_{\omega_I}^k \tag{8}$$

Proof. See Appendix 7.2.

In the next theorem, we show that the EM algorithm for this problem is guaranteed to converge to the maximum likelihood solution which is the solution for (2). Therefore the likelihood value increases slowly as it approaches the optimal solution. Thus a stopping condition for the algorithm is when the likelihood difference between two consecutive steps is sufficiently small.

Theorem 3. *The estimated density function given by the algorithm converges to the maximum likelihood solution $\hat{\Theta}$ defined in the Equation (2).*

Proof. We will first prove that $Q(\Theta|\hat{\Theta}^k)$ is concave in θ_{ω_I}. In Theorem 1, we prove that the value of the auxiliary function $Q(\Theta|\hat{\Theta}^k) = \sum_{\omega_I} \hat{\theta}_{\omega_I}^k \log(\theta_{\omega_I})\phi_{\omega_I}^k$ which is the non-negative linear combination of $\log(\theta_{\omega_I})$. Since $\log(x)$ is a concave in x, the non-negative linear combination of $\log(x)$ functions is also concave. Thus Q is concave in θ_{ω_I}.

Wu et al. [9] showed that the value of the likelihood increases after each iteration. Because Q is concave, the iterative algorithm will finally converge to $\hat{\Theta}$ which maximizes the likelihood function defined in (2).

3 Perturbation of Location and Data

Having presented a general algorithm for reconstruction of community statistics, it remains to decide on the perturbation function. This question is equivalent to choosing the noise probability density function, from which noise samples are chosen. Perturbation is application specific, since it depends on what is being perturbed. We consider the class of applications where we perturb location-tagged data collected by vehicles.

In our application, individuals collect GPS longitude, GPS latitude, speed and (coarsely discretized) time, using their own GPS devices. Once the aggregation server receives perturbed data from participants, the community joint density (i.e., the joint density of longitude, latitude and speed) is reconstructed using the above reconstruction algorithm. Speed-related statistics are then computed as a function of location on the map from the reconstructed joint density. In this paper, we present useful community statistics that can be computed from the estimated multidimensional density such as community average speed, speed distribution, car density, and percentage of speeding vehicles on different streets.

The application was deployed on top of our existing architecture for participatory sensing called PoolView [5]. PoolView is a generic client-server based architecture that enables individuals to collect, archive, and share sensor data with a community On the client side, PoolView provides software that collects sensor data from specific devices (e.g., Garmin GPS). We modified the PoolView client to use our new multidimensional data perturbation scheme. On the server side, we implemented the multidimensional density reconstruction algorithm and the algorithms used to estimate the aforementioned statistics.

3.1 The Perturbation Model

In this section, we propose an algorithm that generates fake (but realistic-looking) vehicle traces that perturb true user location and speed in a way that protects them from being estimated. The vehicle traces are recorded as displacements from an origin (of a coordinate framework) that lies at some agreed upon point in the city in question. These displacements, which we henceforth call *perturbation traces*, will then be added to real routes to generate perturbed routes. There has been many research efforts on generating vehicle traces in prior work [10–13]. We can utilize one of those models to generate perturbation traces for our application. However, the vehicle traces used for perturbation do not need that level of accuracy. Thus, we develop a simplified model that generates perturbation traces using a minimal number of simple parameters.

It is key that the perturbation traces generated resemble real traces for the city in question. For example, in a city with a lot of curvy roads, generated perturbation traces containing only straight segments will not help conceal the identifying characteristics of the roads actually traveled. A robust perturbation trace generation algorithm must therefore incorporate as many features of the actual map as possible.

Our perturbation trace generation algorithm generates traffic routes made of sequences of straight line segments, each of a length drawn from the distribution of the lengths of city blocks. These segments are at angles generated from the distribution of city street intersection angles. This distribution heavily favors 0 degree angles (continuing forward past an intersection) and 90 degree turns. Other angles are generated with lower probability. We ignore U-turns because they occur with a very small probability. For speed, we use a sine curve for each road segment that peaks in the middle of the segment and slows down towards the beginning and end. The peak is drawn from the distribution of city street speed limits. The slowest point is a uniformly-distributed random fraction of the peak. These traces represent displacement to actual routes. This displacement can be scaled to control the noise variance.

Finally, for the purpose of reconstructing the community joint distribution, we need the joint distribution of the generated perturbation trace (the noise). Since it is hard to come up with an analytic solution for the joint distribution of the noise, we generate this distribution numerically. First, we generate a pool of noise data points from the model then *a non-parametric density estimation with smoothing* [14] is employed to estimate the joint distribution. In this application, 5000 vehicle traces, each of which contains 40 data points, are generated and used as input to the density estimation algorithm, which generates the joint distribution.

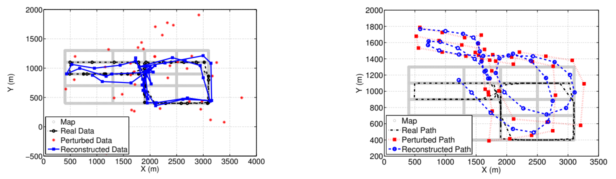
3.2 Achieved Privacy

In this section, we analyze the extent of privacy offered to *individual* user data using our perturbation scheme. The information available to the aggregation server includes the perturbed data, the noise density function (known by the server) and the map on which the user traveled. First, note that the reconstruction algorithm proposed in this paper can not be used to reconstruct individual's real data from those information. Our proposed algorithm can only reconstruct community distribution from shared data of a reasonable number of participants. Using the available information, the malicious server can employ filtering techniques to remove additive noise from the perturbed data. We call this kind of attack *filtering attack*.

In this paper, we analyze a filtering attack which applies a Wiener filter to remove additive noise from perturbed data. The Wiener filter uses the noise density information to filter the noise from perturbed data. One important assumption that the Wiener filter makes is the noise samples are independent. However, this assumption fails because the noise samples generated by our algorithm are correlated which makes the estimated data traces follow the perturbed path instead of real path. For demonstration, we perturb a real user location trace with both correlated noise generated by our algorithm and independent Gaussian white noise and then perform the Wiener filter on both perturbed data set.

The result of the Wiener attack in the case of Gaussian white noise is shown in Figure 1(a). The reconstructed path is very close to the real path and the reconstruction error is less than one block which means that the attacker can easily figure out the place where the user have been. Figure 1(b) shows the real path, perturbed path and the reconstructed path for the perturbation technique we developed in this paper. We see that the reconstructed path follows the perturbed path. Therefore, the Wiener filter attack does not work as desired for the attacker. Users might want to increase the variance of the generated noise to get more privacy, but the reconstruction error might increase as well. Therefore, it is important to balance the trade off between privacy and accuracy.

The second type of attack considered in this paper is the range attack. It is possible to conduct the range attack in applications where the ranges of both the real data and the generated noise are finite. In this case, real data values can be inferred if boundary values of the perturbed data are observed. For example, suppose the real speed of a vehicle is in the range [0 to 50] and the generated noise is also in the range [0 to 50]. If the perturbed speed is 100, the attacker knows with certainty that the true speed is 50. In general, if the perturbed values are close to the boundary, privacy can be violated. In applications involving GPS location as a private variable, however, this attack is not effective. GPS location refers to a point of the globe. Perturbing that location by a few miles is sufficient for privacy, yet the perturbed location still refers to a point on the globe. In other words, the perturbed coordinates always refer to a valid data



(a) Reconstruction of user location with Gaussian noise

(b) Reconstruction of user location with correlated noise

Fig. 1. Reconstruction of user location perturbed with different noise model.

point. An exception is when map information is used to infer noise. For example, at coastal areas, one may safely assume that vehicles do not move on water, which generates a boundary on valid locations. The map-based attack will be discussed shortly. In general, the effect of range-based attacks can be mitigated if the noise distribution has a long tail such that arbitrarily large values are allowed with an arbitrarily low probability. (Many distributions, including Gaussian, have this property.) In this case, the range is infinite. There is no maximum value for the perturbed signal that can be used to breach privacy.

Another popular type of attack against additive-noise perturbation techniques is the *leak attack* [15]. In this type of attack, the attacker may be able to estimate the seed of the pseudo random number generator which generates the noise curve if he can guess a few true data values. Then this seed can be used to generate the noise curve used by the user since the noise distribution is known. However, with our perturbation scheme, this attack is not possible because we only use the random number generator to generate the model parameters (e.g., number of turns, speed of each segment). The additive noise is then generated using those parameters and the model developed earlier in this section.

A vulnerability of our perturbation scheme is that it is possible to combine the real map with a clever estimation technique to estimate the most likely traveled path. We call this attack scheme a *map-based attack*. At this moment, it is still unknown if there exists a good map-based attack against our perturbation scheme. In this paper, we argue that finding an efficient map-based attack is hard. One possible way to conduct the map-based attack is to look at the sequence of the turning angles in the GPS trajectory data. Since the probability that the noise angle and the real angle cancel out is pretty small, the turning angles from the perturbed data contain some information about the real turning angles. Combining with the map, it is possible to find the most probable traveled path. It is not easy, however, to find the likelihood of the real turning angle given the perturbed path. Because the perturbed path is created by adding the coordinates of the real path and the noise path, the angle in the perturbed path is not only depend on the angle of both real path and noise path but also depend on the magnitudes of those. In the upcoming sections, we only evaluate the immunity of our perturbation scheme against filtering attacks.

4 Simulation Results

In this section, we evaluate the performance of the traffic mapping application with simulated data. The advantage of using simulated data is to give total control over traffic parameters, (e.g., average community speed, speed map), which is hard to accurately measure in a real application. In addition, vehicular traces can be generated for a large numbers of "virtual" users makes it possible to evaluate the accuracy of the reconstruction algorithms. We also evaluate the accuracy computation of the community average speed using the reconstructed density in this section.

We use the ONE (Opportunistic Network Environment) [16] simulator to generate artificial traces of vehicle movements in a small city setup. The map used in this simulation is a part of Helsinki city and is distributed with the ONE simulator. The simulator supports Map Based Movement models that can import map data and constrain vehicle movement to the streets and roads of the imported map.

Our goal is to make the data get out from the simulator as realistic as possible. The input map for the simulator is extracted from a real map and is shown in Figure 2 with the X and Y coordinates ranging from 0 to 4000 meters and 0 to 3600 meters respectively. Vehicle speeds are chosen to be Gaussian with mean 30mph and standard deviation of 10mph. Trip data, including X and Y coordinates and vehicle speed, are sampled at a frequency of 1 Hz, and are stored in an external file for later use. The simulated data are then perturbed with perturbation traces generated by the algorithm discussed in Section 3.1. The perturbed data are then submitted to the aggregation server.

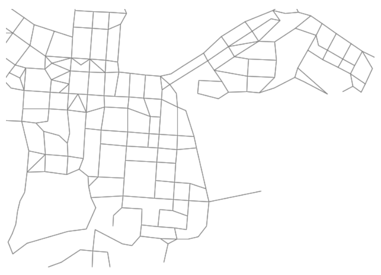


Fig. 2. The map used in simulation

We collect data from 120 users, each of which contains 80 data points, from the simulation. In order to reconstruct the community joint distribution, we first have to specify the range of each dimension and the number of bins in each dimension. Those parameters are summarized in Table 1. In this simulation, we discretize the location in 100mx100m bins which is small enough to capture the street information. For more accurate reconstruction of the joint density, more bins in each dimension might be needed but it would require more user data points and computational time. In this specific traffic application, we are only interested in the density values corresponding to the street locations. Our proposed algorithm allows us to do the reconstruction on those bins only thus siginificantly reduce the time complexity of the algorithm.

Table 1. Parameters for the reconstruction

Parameter	range of X	range of Y	range of V
Value	0 - 4000 (m)	0 - 3600 (m)	0 - 60 (mph)
Parameter	X bins	Y bins	V bins
Value	40	36	60

Table 2. Noise variance in each data set

Parameter	stddev of X (m)	stddev of Y (m)	stddev of V (mph)
Dataset 1	100	100	4
Dataset 2	500	500	36
Dataset 3	900	900	60
Dataset 4	1500	1500	76
Dataset 5	3000	3000	100

In the first experiment, we study the accuracy of the density reconstruction algorithm under various noise variance. The application must achieve high reconstruction accuracy at a reasonably high noise variance level in order to provide sufficient privacy to users. To achieve this goal, we perturbed the simulation data using five different noise variances shown in Table 2.

We define the accuracy of the density reconstruction as a function of the average accuracy of all the bins:

$$r = \frac{1}{K} \sum_{i=1}^{K} \left(1 - \frac{|\theta_i - \hat{\theta}_i|}{\theta_i} \right) \tag{9}$$

In Equation (9), r is the computed accuracy, θ_i is the true discrete density parameter, $\hat{\theta}_i$ is the estimated density parameter. $\hat{\theta}_i$ is obtained by feeding the real density using real user data points to the density estimation algorithm.

The accuracies of the reconstructions as the function of the number of data points and noise variance are shown in Figure 3. The figure shows five different curves corresponding to the five dataset described above. The X axis is the number of data points which varies from 120 points to 1200 points with 120-point increments. In the results, Dataset 1 achieves highest accuracy while Dataset 5 achieves lowest accuracy.

Next, we evaluate the achieved privacy for each dataset presented in Table 2. We assume that the attacker uses Wiener filter to estimate vehicle trace of *individuals* from perturbed data and the noise distribution. Beside correlated noise, trip data are also perturbed with Gaussian noises with the same standard deviation for comparision purpose. We perform the estimation on the perturbed vehicle trace of all users and compute the average reconstruction error which is presented in Table 3 below.

From the Table 3, the reconstruction error for the vehicle traces perturbed with correlated noise is very high as opposed to the Gaussian case in which the

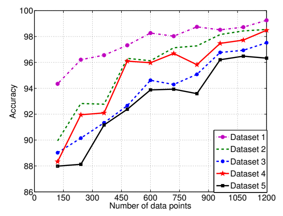


Fig. 3. Percentage reconstruction accuracy as a function of number of data points and noise variance

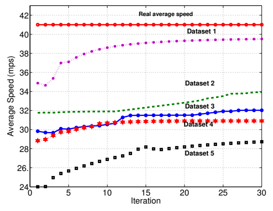


Fig. 4. Community average speed versus number of iterations

Table 3. Reconstruction Error of Individual Data

Dataset	Correlated Noise (m)	Gaussian Noise (m)
Dataset 1	334.5	145.0
Dataset 2	1329.5	153.4
Dataset 3	1942.4	189.8
Dataset 4	3573.6	218.1
Dataset 5	4901.1	223.5

error is small. With Dataset 1 (the noise covariance is small) the reconstruction of individual data is still high (about 3 blocks) which means good privacy is achieved. Also, with Dataset 5, although the reconstruction error of individual data is huge (about 40 blocks), the community distribution can still be accurately reconstructed (above 96%).

In the last experiment, we demonstrate the estimation of the community average speed using the joint distribution estimated in the first experiment. In addition, we also want to study the effect of the number of iterations on the accuracy of reconstruction. To compute the community average speed from the community joint distribution $f(X, Y, V)$, we first compute the speed density $f(v)$

$$f(v) = \sum_{x=1}^{40} \sum_{y=1}^{36} f(x, y, v) \Delta_{XY} \tag{10}$$

Equation (10) is the marginalization of the discrete joint density over X and Y dimensions. where $\Delta_{XY} = (4000/40) * (3600/36)$ is the area of a two dimensional bin XY. Then the average speed $\bar{v}$ is computed as $\bar{v} = \sum_{v=1}^{60} v f(v)$.

The result of the experiment is shown in Figure 4. Although Dataset 5 provides users with highest acceptable privacy, the reconstructed average speed is still close to the true value. Another important observation from the graph is that the density reconstruction algorithm requires a very small number of iterations to converge. Results from 5 datasets show that 10 to 15 iterations are sufficient. The accuracy of the algorithm almost does not change after 20 iterations. In the next section, we evaluate the performance of the application using deployment data.

5 Deployment Data

In this section, we evaluate the traffic monitoring application with real deployment data. The data are collected by driving on all the streets within an area shown in Figure 5. There are a total of 15 users, each user drives the streets at will for 10 minutes. During the drive, we use a Garmin Legend [17] GPS device to record location and speed information. The sampling frequency of the device is 15Hz which is enough to record changes in the location and speed since the speed limit in the area is 25 mph.

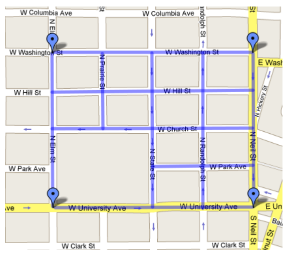


Fig. 5. Map used to collect data

At the aggregation server side, to do the reconstruction, we need to specify the reconstructed region and the number of bins in each region. The reconstruction parameters are summarized in Table 4. For location, we divide each axis into 30 bins, the width of each bin is 0.01 mile, which is about the width of a street. This is important because, we want to estimate the speed down to the resolution of a street. This can be done by looking at the specific bins corresponding to the target street.

Table 4. Parameters for the reconstruction

Parameter	range of X) (1/100 mile)	range of Y (1/100 mile)	range of V (mph)
Value	0 - 300	0 - 300	0 - 25
Parameter	X bins	Y bins	V bins
Value	30	30	30

Table 5. Noise standard deviation

Parameter	stddev of X (1/100 mile)	stddev of Y (1/100 mile)	stddev of V (mph)
Dataset 1	45	35	5
Dataset 2	75	75	10
Dataset 3	100	100	15
Dataset 4	150	150	20
Dataset 5	300	300	30

In the first experiment, we study the density reconstruction accuracy as a function of the number of data points used for reconstruction. We want to answer the question of how many data points we need to achieve a desired accuracy. Similar to the case of simulation data, we do the perturbation of the data with five different noise data sets each of which has different variance. The details of the noise datasets are presented in Table 5. The standard deviation of the noise specified in the table is comparable to multiples of the block length (about 75/100 mile), We run the density reconstruction algorithm multiple times, each time with a different number of data points. The data points are randomly picked from the total pool of data points contributed by all users. The number of data points taken for reconstruction is varied from 100 to 800.

The results of the experiment are shown in Figure 6. From the result, the highest accuracy achieved is about 90% at about 800 datapoints while the lowest accuracy is about 83% at about 160 datapoints. The number of data points needed for a good estimate is thus surprisingly low. This can be explained by the observation that since the data points are uniformly picked from the pool, there is a high chance that they scatter all over the map, thus capturing the speed information of the whole area. This makes the application practical in most city areas. In the next experiment, we demonstrate the estimation of the community speed distribution. This community speed distribution can be useful in determining the average speed in the area or compute the percentage of speeding vehicles in that area. To compute the community speed distribution $f(v)$, we marginalize the estimated discrete joint distribution $f(x, y, v)$ as follow

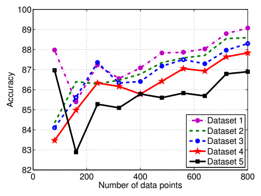
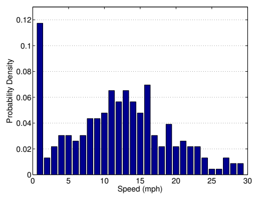


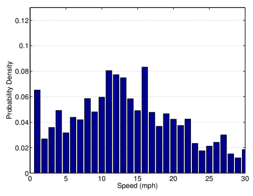
Fig. 6. Accuracy of the density reconstruction

$$f(v) = \sum_{x=1}^{30} \sum_{y=1}^{30} f(x, y, v) \Delta_{XY} \tag{11}$$

where $\Delta_{XY} = (300/30) * (300/30)$ is the area of a two dimensional bin in XY dimension. Figure 7(a) and 7(b) shows the real community speed distribution and the estimated community speed distribution, respectively. We see that the two speed distributions are similar except for the first bin corresponding to zero speed. This can be explained because the density estimation algorithm tends to produce a smooth distribution. Thus, the speed value of the bin is smoothed out. The percentage of speeding vehicles in the community can be computed as the sum of bins with larger than 25 miles/hr speed. In this case the real community percentage of speeding is about 7% while the estimated percentage of speeding is 10% which is a good estimate.



(a) Real community speed distribution



(b) Reconstructed community speed distribution

Fig. 7. Real and reconstructed speed distribution.

6 Conclusion

In this paper, we present theoretical foundations for perturbation based mechanisms for ensuring privacy while allowing correct reconstruction of community statistics of interest. Previous data perturbation techniques fail to ensure either privacy or correct reconstruction of community statistics in the case of correlated multidimensional time-series data. The algorithms proposed in this work allow participants to add noise to multiple correlated data streams prior to sharing in a privacy-preserved way while making sure that relevant community statistics are still reconstructible. A participatory sensing application for traffic monitoring is developed which allows participants to "lie" about their actual location and speed, while letting the community estimate useful traffic statistics (e.g., speed map, percentage of speeding vehicle, etc) with high accuracy.

References

1. Burke, J., et al.: Participatory sensing. In: Proc. of ACM SenSys. (2006)
2. Hull, B., et al.: Cartel: a distributed mobile sensor computing system. In: Proc. of SenSys. (2006) 125–138
3. Eisenman, S.B., et al.: The bikenet mobile sensing system for cyclist experience mapping. In: Proc. of SenSys. (2007) 87–101
4. Miluzzo, E., et al.: Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In: Proc. of SenSys. (2008) 337–350
5. Ganti, R.K., Pham, N., Tsai, Y., Abdelzaher, T.F.: Poolview: Stream privacy for grassroots participatory sensing. In: Proc. of SenSys. (2008) 281–294
6. Agrawal, R., Srikant, R.: Privacy preserving data mining. In: Proceedings of the ACM SIGMOD. (2000) 439–450
7. Agrawal, D., Aggarwal, C.C.: On the design and quantification of privacy preserving data mining algorithms. In: Proc. of ACM SIGMOD. (2001) 247–255
8. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of Royal Statistical Society **B39** (1977) 1–38

9. Wu, J.: On the convergence properties of the em algorithm. The Annals of Statistics **11**(1) (1983) 103, 95
10. Lian, F.L., Murray, R.: Real-time trajectory generation for the cooperative path planning of multi-vehicle systems. In: Proceedings of the 41st IEEE Conference on Decision and Control. Volume 4. (2002) 3766–3769
11. Saha, A.K., Johnson, D.B.: Modeling mobility for vehicular ad-hoc networks. In: Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks, ACM (2004) 91–92
12. Karnadi, F., Mo, Z.H., chan Lan, K.: Rapid generation of realistic mobility models for vanet. In: IEE Wireless Communications and Networking Conference. (2007) 2506–2511
13. Fiore, M., Harri, J., Filali, F., Bonnet, C.: Vehicular mobility simulation for vanets. In: Simulation Symposium, 2007. ANSS '07. 40th Annual. (2007) 301–309
14. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2006)
15. Kelsey, J., Schneier, B., Wagner, D., Hall, C.: Cryptanalytic attacks on pseudorandom number generators. In: FSE '98: Proceedings of the 5th International Workshop on Fast Software Encryption, London, UK, Springer-Verlag (1998) 168–188
16. `http://www.netlab.tkk.fi/tutkimus/dtn`
17. `www.garmin.com/products/etrexLegend`

7 Appendix

7.1 Proof of Theorem 1

We begin with the expansion the auxiliary function Q by noting that the data points are i.i.d.

$$\begin{aligned} Q(\Theta|\hat{\Theta}^k) &= E_{X|Y}\left[\log f_{X;\Theta}(\bar{X})|\bar{Y},\hat{\Theta}^k\right] \\ &= E_{X|Y}\left[\log \prod_{j=1}^{N} f_{X;\Theta}(x_j)|y_j,\hat{\Theta}^k\right] \\ &= \sum_{j=1}^{N}\int_{\Omega}\log f_{X;\Theta}(\gamma) f_{X|Y;\hat{\Theta}^k}(\gamma|y_j)d\gamma \end{aligned}$$

In the last step, the expectation is taken over all possible values of X given the observation y_i. We further expand the auxiliary function Q using Bayes' formula and the fact that $f_{Y|X}(Y|X) = f_N(Y-X)$ because $N = Y - X$.

$$\begin{aligned} Q(\Theta|\hat{\Theta}^k) &= \sum_{j=1}^{N}\int_{\Omega}\log f_{X;\Theta}(\gamma)\frac{f_{XY;\hat{\Theta}^k}(\gamma,y_j)}{f_{Y;\hat{\Theta}^k}(y_j)}d\gamma \\ &= \sum_{j=1}^{N}\frac{1}{f_{Y;\hat{\Theta}^k}(y_j)}\int_{\Omega}\log f_{X;\Theta}(\gamma) f_{X;\hat{\Theta}^k}(\gamma) f_N(y_j-\gamma)d\gamma \\ &= \sum_{j=1}^{N}\frac{1}{f_{Y;\hat{\Theta}^k}(y_j)}\sum_{\omega_I}\int_{\omega_I}\log(\theta_{\omega_I})\hat{\theta}^k_{\omega_I} f_N(y_j-\gamma)d\gamma \end{aligned}$$

In the last equation, the integral over the Ω is discretized and is computed as the sum of the integral over all subspaces ω_I in which the value of the discrete density function is constant. Also the value of $f_{Y;\hat{\Theta}^k}(y_j)$ is computed as follow:

$$\begin{aligned} f_{Y;\hat{\Theta}^k}(y_j) &= \int_{\Omega} f_Y(y_j|x) f_{X;\hat{\Theta}^k}(x)dx \\ &= \sum_{\omega_I} f_N(y_j - \omega_I)\hat{\theta}^k_{\omega_I} \end{aligned}$$

$$\begin{aligned} Q(\Theta|\hat{\Theta}^k) &= \sum_{j=1}^{N} \frac{1}{f_{Y;\hat{\Theta}^k}(y_j)} \sum_{\omega_I} \hat{\theta}^k_{\omega_I} \log(\theta_{\omega_I}) \int_{\omega_I} f_N(y_j - \gamma)d\gamma \\ &= \sum_{\omega_I} \hat{\theta}^k_{\omega_I} \log(\theta_{\omega_I}) \sum_{j=1}^{N} \frac{f_N(y_j - \omega_I)}{f_{Y;\hat{\Theta}^k}(y_j)} \\ &= \sum_{\omega_I} \hat{\theta}^k_{\omega_I} \log(\theta_{\omega_I}) \phi^k_{\omega_I} \square \end{aligned}$$

7.2 Proof of Theorem 2

This is an optimization problem with a constraint which ensures that Θ is a proper density function.

$$\hat{\Theta}^{k+1} = \underset{\Theta}{\operatorname{argmax}}\, Q(\Theta|\hat{\Theta}^k)$$

$$\sum_{\omega_I} \theta_{\omega_I} m_{\omega_I} - 1 = 0$$

The Lagrangian of the optimization is given by

$$\begin{aligned} L(\theta_{\omega_I}, \lambda) &= Q(\Theta|\hat{\Theta}^k) + \lambda(\sum_{\omega_I} \theta_{\omega_I} m_{\omega_I} - 1) \\ &= \sum_{\omega_I} \hat{\theta}^k_{\omega_I} \log(\theta_{\omega_I}) \phi^k_{\omega_I} + \lambda(\sum_{\omega_I} \theta_{\omega_I} m_{\omega_I} - 1) \end{aligned}$$

The optimized values $\hat{\theta}^{k+1}_{\omega_I}$ satisfied $\frac{\partial L}{\partial \theta_{\omega_I}}(\hat{\theta}^{k+1}_{\omega_I}) = 0$ and $\frac{\partial L}{\partial \lambda}(\hat{\theta}^{k+1}_{\omega_I}) = 0$. After some algebraic transformation we get

$$\lambda = -\frac{1}{N} \sum_{j=1}^{N} \frac{1}{f_{Y;\hat{\Theta}^k}(y_j)} \sum_{\omega_I} \hat{\theta}^k_{\omega_I} f_N(y_j - \omega_I) \tag{12}$$

Since $Y = X + N$ thus the density of Y is the convolution of the density of X and N. It is straight forward to show that

$$f_{Y;\hat{\Theta}^k}(y_j) = \sum_{\omega_I} \hat{\theta}^k_{\omega_I} f_N(y_j - \omega_I) \tag{13}$$

Substitute (13) into (12) yield $\lambda = -1$. Therefore

$$\hat{\theta}^{k+1}_{\omega_I} = \frac{\phi^k_{\omega_I}}{m_{\omega_I}} \hat{\theta}^k_{\omega_I} \square$$