OptiMoS: Optimal Sensing for Mobile Sensors

Zhixian Yan*, Julien Eberle†*, Karl Aberer*
* EPFL, Switzerland
† Nokia Research Center, Lausanne, Switzerland
E-mail: zhixian.yan@epfl.ch, julien.eberle@nokia.com, karl.aberer@epfl.ch

***Abstract*—Both *sensor coverage maximization* and *energy cost minimization* are the fundamental requirements in the design of real-life mobile sensing applications, e.g., (1) deploying environmental sensors (like CO_2, fine particle measurement) on public transports to monitor air pollution, (2) analyzing smartphone embedded sensors (like GPS, accelerometer) to recognize people daily activities. However sensor coverage and energy cost contradict each other: the higher frequency mobile sensing takes, the more energy is used; and vise versa.**

In this paper, we design a novel two-step mobile sensing process ("OptiMoS") to achieve optimal mobile sensing that can effectively balance sensor coverage and energy cost. In the first step, OptiMoS divides the continuous mobile sensor readings into several segments, where the readings in one segment are highly-correlated rather than readings amongst different segments. In the second step, OptiMoS identifies optimal sampling for the sensor readings in each segment, where the selected readings can guarantee reasonably high sensor coverage with limited sampling rate. Various greedy & near-optimal *segmentation* and *sampling* methods are designed in OptiMoS, and are evaluated using real-life environmental data from mobile sensors.

I. Introduction

Wireless sensor networks (WSN) and publishing of sensor data on the Internet bear the potential to substantially increase public awareness and involvement in environmental sustainability. Air quality monitoring in urban areas is a prime example of these applications as common air pollutants have direct effects on the human health. Traditional WSN based environmental monitoring systems (like SensorScope [12] and MacroScope [26]) typically deploy sensors on some pre-selected positions, monitor these fixed sensors, and analyze their measurements. These traditional environmental monitoring systems based on static WSN using fixed sensors have a couple of obvious limitations, e.g.,: (a) the system is inflexible for monitoring location-varying environment as the sensor placement is pre-fixed; (b) it is expensive to deploy and maintain a large set of static sensors; (c) for monitoring a very large area, it is impossible to get enough static sensors to cover the complete area.

To overcome such limitations of using static WSN for environmental sensing, researchers recently start to build WSN with mobile sensors. There are a lot of emerging mobile sensing platforms, e.g.,: (a) the OpenSense project in Switzerland builds sensors and deploys them on public transports like buses and trams [2]; (b) the floating sensor network project at UC Berkeley builds a water monitoring system using drift sensors to analyze water contaminant [1]; (c) mobile phones are used to establish a community seismic network to detect earthquakes and rare events [9]. These projects use mobile sensors and bring public involvement in environmental monitoring to a reality, which poses today substantial research and technical challenges for the communication and information systems infrastructure, to scale up from isolated well controlled systems to an open and scalable infrastructure.

For traditional static WSN applications (e.g., environmental monitoring and others), we need to find the best places to fix these limited sensor resources (e.g., the number of sensors) that can maximize sensor coverage. This is the fundamental sensor placement problem of WSN; and a good placement can guarantee the coverage and reflects how well an area is monitored by sensors. Determining an optimal sensing placement in an arbitrary sensor field is a kind of *art gallery* problem, which is NP-hard [18] and requires near-optimal solutions like the submodularity method [15].

Regarding mobile sensors based environmental monitoring, sensor placement is even more challenging compared to static WSN; this is because mobile sensors have varying coverage due to their mobility. In the OpenSense project, we deploy environmental sensors on moving buses to monitor air quality (using sensors to measure CO_2, CO, NO_2 etc.), which are mobile sensors as the buses are regularly running on the road. In OpenSense, there are two levels of objectives for optimal sensor placement: the first one is to build the optimal mobile sensing for each individual bus line, i.e., determining when and where the bus should take a measurement; the second one is to build global optimal sensing, which provides optimal sensing for all bus lines, where individual sensing on one bus line should consider nearby bus lines. For example, Fig. 1 shows optimal mobile sensing of four bus lines, where the times symbol ("$\times$") indicates the sensing points when and where the bus pass by. In such mobile sensing planning, $Bus\ 3$ only has two sensing points (blue "$\times$"), as there are several overlapping sensing points with other buses. In this paper, we focus on the first mobile sensing objective of OpenSense, i.e., the optimal sensing strategy for each individual moving bus.

Research Questions: We seek to design an optimal mobile sensing strategy, which offers an appropriate tradeoff between "sensing coverage maximization" and "sensing cost (sampling) minimization" of an individual moving sensor. This raises two key research questions:

a) For a short sequence of mobile sensing (e.g., a moving bus in 2 hours), how to find an optimal sensing (or sampling) protocol which can guarantee required sensor coverage using limited sensing cost (or sampling rate)?

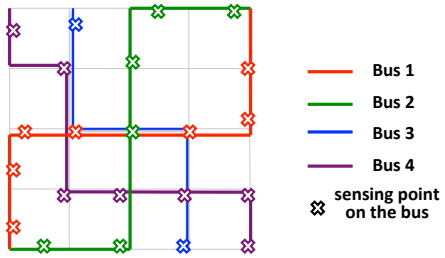

Fig. 1: Mobile sensing via moving buses with environmental sensors

b) For a long sequence of mobile sensing (e.g., a moving bus in one month or several days), how to find optimal sensing? Do we need to do segmentation first? Can segmentation help to make better sensing (sampling)?

Key Contributions: To address these questions, this paper utilizes real-life mobile sensing data of environmental monitoring from the OpenSense project. We propose a model-driven optimal sensing strategy for mobile sensors. The detailed contributions of this paper are as follows:

a) We design "OptiMoS", a two-tier optimal sensing framework for mobile sensors, using model-driven data regression (e.g., linear model, support vector regression). In the first tier, a long sequence of mobile sensing (e.g., the complete route of a bus line) is divided into several non-overlapping segments, where the data points in each segment are homogeneous in terms of modeling and prediction. In the second tier, OptiMoS chooses optimal sampling points for each segment, as these data points can provide maximum sensor coverage.

b) For segmenting sensing data from mobile sensors, we design and compare several different segmentation algorithms, from the optimal exhaustive search using dynamic programming, to the binary top-down segmentation, to the near-optimal error-based heuristic segmentation.

c) For sampling the sensor readings in each segment, we design and compare several sampling algorithms, including the uniform, random, error-based entropy sampling, and the near-optimal mutual-information sampling.

d) To validate such optimal mobile sensing method of OptiMoS, all of these segmentation and sampling algorithms are exhaustively evaluated using real-life environmental sensing data from the moving buses.

The detailed structure of this paper is organized as follows: after the introduction, Section II summarizes existing related works; Section III describes the preliminaries and the two-tier optimal sensing framework of OptiMoS; Section IV presents different segmentation algorithms for dividing mobile sensing stream; whereas Section V proposes different sampling strategies for individual segment; in Section VI, we experimentally evaluate the OptiMoS. Finally, Section VII includes concluding remarks and points to future works.

II. Related Work

There are three main topics related to this paper: (1) optimal sensor placement in static WSN; (2) mobile sensing; and (3) sensor data segmentation.

Sensor Placement in WSN: Sensor placement has a significant influence in building an efficient wireless sensor network for environment monitoring. An optimal sensor placement should be able to maximize the sensor coverage and in the meanwhile to minimize the number of sensors required. For placing K sensors optimally in an arbitrary sensor field, this work is known to be NP hard; thus a couple of efficient near-optimal algorithms are provided [17][20][7][21]. In Andreas' works like [17], a greedy solution is designed by using mutual information when selecting next sensor point; this has better performance than traditional random solution or other basic entropy-based sensor selection. In [20], the sensor placement is modeled as a min-max optimization problem, and a simulated annealing based algorithm is provided. In [7], the method supports imprecise sensor detection like terrain properties, which can support sensor placement with probabilistic guarantee in a polynomial time. The work of regions sampling in [21] studies local aggregation on sensor network and builds adaptive sampling for each partial region. All of these works are not designed for mobile sensors. Nevertheless, relevant optimization formulation and greedy solutions can be adopted in OptiMoS, to achieve optimal sensing of an individual moving bus with deployed environmental sensors.

Sensing from Mobile Sensors: Recently, there has been emerging a number of mobile sensing applications, particularly in urban computing, such as OpenSense for air quality monitoring [2], floating sensors for water contaminants analytics [1], and community sensing for road traffic monitoring [16]. One recent work similar to our optimal sensing objective in OpenSense is the Ear-Phone [25], which provides an end-to-end participatory urban noise mapping system and generates a noise map from a small set of sensor readings at a sparse spatio-temporal sensing field. Similarly, our objective in OpenSense is to provide a time-varying air pollution map from limited mobile sensor readings using a small number of moving buses with deployed environmental sensors (e.g., CO_2). However, our work in OpenSense has additional challenges: (a) the monitoring area in OpenSense is not 1D road line like [25], but a 2D map and even 3D earth; and (b) OpenSense is real mobile sensing that deploy sensors on the moving buses, whilst Ear-Phone fixes mobile phones aside the road.

Sensor data segmentation: Segmentation is largely studied in time series [14][13][11] and GPS-alike mobility data [27][5]. Segmentation methods are divided into three categories, i.e., *sliding window*, *top-down* and *bottom-up*. Sliding window algorithms can work online and efficiently, but the results are poor and sensitive to parameters; whilst *top-down* and *bottom-up* methods have better segmentation results but not for real-time applications. To balance both offline high-accuracy and online efficient-computation, a couple of combination algorithms are proposed, such as SWAB (Sliding Window And Bottom-up) [13], amnesic functions [23], piecewise linear segmentation (mixing both constant and linear function) [19], FSW (Feasible Space Window) & SFSW (Stepwise FSW) [22], SwiftSeg (a polynomial approximation of a time series in either sliding or growing windows) [10]. In this paper,

we study different segmentation methods, test them with the combination of different sampling strategies, and evaluate the optimal sensing proposal of OptiMoS.

III. Two-Tier Optimal Mobile Sensing

This section presents our two-tier optimal mobile sensing proposal (i.e., OptiMoS) and its problem formulation. Fig. 2 sketches the framework of OptiMoS for achieving the optimal sensing, as well as the data flow in this procedure.

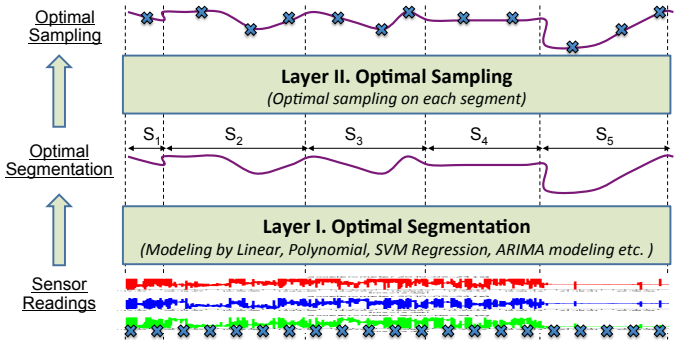

Fig. 2: OptiMoS's two-tier optimal sensing framework

In the lower tier of OptiMoS, initial input is the raw sensor readings collected by moving sensors, i.e., multiple dimensional spatio-temporal time series data. Each reading record is the "×" symbol in Fig. 2 (i.e., $R_i = \langle t, l, x_1, \cdots, x_m \rangle$), which includes sensing time t, sensing location l (typically $\langle longitude, latitude \rangle$ from GPS), and environmental measurements x_1 to x_m[1]. The objective of this tier is to find the optimal (or near-optimal) segmentation based on data modeling on these raw readings. OptiMoS can support all kinds of modeling methods, e.g., simple linear regression, polynomial regression, SVM (Support Vector Machine) based regression, time series ARIMA (Auto-Regressive and Moving Average) modeling. As the result of the first tier, we can achieve an optimal (or near optimal) segmentation, e.g., five segments (from S_1 to S_5 in Fig. 2).

In the upper tier of OptiMoS, we focus on studying individual segments that are computed from the lower tier. For each segment, the objective is to find the best sampling from the mobile sensor readings, i.e., to select only a subset of sensor readings ("×" symbols in Fig. 2 in the top layer). This subset can keep enough modeling information for regression of the whole segment and for prediction of non-selected sensor readings. Taking Fig. 2 for example, from segment S_1 to S_5, we respectively keep only 1, 3, 3, 2, 3 readings to achieve reasonably enough sensing. For this optimal sensing example, OptiMoS only requires 12 sensing points instead of the initial 21 points. The sampling rate is 12/21, i.e., 57%.

A. Problem Statement

As mentioned, the OpenSense project has two levels of optimal sensing objectives, i.e., local optimal sensing for individual bus line, and global optimal sensing for multiple bus lines. In this paper, we focus on the first objective and design OptiMoS (the two-tier optimal sensing) for mobile sensors in terms of a single bus line.

This optimal sensing problem can be formulated as follows: *Given a sequence of initial mobile sensor readings* $\mathcal{R} = \{R_1, \cdots, R_N\}$ *of size* N *from continuously moving sensors, where each record* $R_i = \langle t, l, x_1, \cdots, x_M \rangle$ *consists of* M *types of sensor readings (from* x_1 *to* x_M*) together with the timestamp* (t) *and the location* $(l = \langle longitude, latitude, altitude \rangle)$*, the objective of OptiMoS is to identify the best sampling of such sequence of sensor readings that can guarantee the majority of sensor reading information (i.e., sensor coverage maximization) at the minimum sampling rate (i.e., energy cost minimization).*

As shown in Fig. 2, our solution for this problem is to provide a two-tier optimization framework. We will compare this with traditional one-tier sampling without segmentation, and further details will be provided in Section VI.

B. Optimal Segmentation

As the first tier of OptiMoS, optimal segmentation on the initial reading sequence (i.e., $\mathcal{R}$) is to find the best K segments (i.e, $\mathcal{R}_1, \mathcal{R}_2, \cdots, \mathcal{R}_K$) such that the sum of the model errors for individual segment is minimized. A model $\mathcal{M}$ on a segment $\mathcal{R}_i$ can be linear, polynomial, SVM regression, ARIMA etc. In this paper, we empirically study linear and SVM regression, and evaluate their performances; other models could have similar principle. We apply the RSS (*Residual Sum of Squares*) to quantify the error for modeling a sequence $\mathcal{R} = \{R_1, R_2, \cdots, R_N\}$ (see Formula 1). The residual $res(R_i)$ of a reading R_i is the difference between real value in R_i and the approximation $\hat{R}_i$ that learnt by the model $\mathcal{M}(\mathcal{R})$.

$$RSS(\mathcal{M}(\mathcal{R})) = \sum_{i=1}^{N} (res(R_i))^2 = \sum_{i=1}^{N} (|R_i - \hat{R}_i|)^2$$
$$\text{where } \hat{R}_i = \mathcal{M}(\mathcal{R})|_{R_i} \quad (1)$$

Finding the best K segments is equivalent to identifying the best K-1 division points: $R_{d_1}, R_{d_2}, \cdots, R_{d_{K-1}}$; then, for each segment $\mathcal{R}_i$, we have a sub sequence of readings between two division points, i.e, $\mathcal{R}_i = \{R_{d_{i-1}}, R_{d_{i-1}+1}, \cdots, R_{d_i}\}$. For the first segment ($\mathcal{R}_1$), R_{d_0} indicates the first reading R_1. Now, this optimal segmentation problem becomes an unconstrained optimization problem (see Formula 2).

$$\underset{d_1, d_2, \cdots, d_{K-1}}{argmin} \sum_{i=1}^{K} RSS(\mathcal{M}(\{R_{d_{i-1}}, \cdots, R_{k_{d_i}}\})) \quad (2)$$

Ideally, the segment number (K) is not known in advance, which needs to be discovered automatically as a part of the optimization problem, as shown in Formula 3.

$$\underset{K, d_1, d_2, \cdots, d_{K-1}}{argmin} \sum_{i=1}^{K} RSS(\mathcal{M}(\{R_{d_{i-1}}, \cdots, R_{k_{d_i}}\})) \quad (3)$$

[1]Taking air quality monitoring using environmental sensors for example, x_1 is the measurement of CO_2, x_2 is of CO, x_3 is of NO_2, etc.

For simplicity, in the first step of this work, we can assume K is given and we will test a reasonably small set of different K values (e.g., $K \leq 10$ in our experiment), to analyze the convergence of segmentation algorithms and to test a near-optimal segmentation.

C. Optimal Sampling

After getting the optimal segmentation, OptiMoS needs to identify the best sampling of mobile sensing/readings for each individual segment. To quantify whether a sampling reading sequence $\mathcal{R}_{sub}$ is good or not, we define "*information loss*" $\mathcal{L}(\mathcal{R}, \mathcal{R}_{sub})$, i.e., the RSS increase ratio between $\mathcal{R}_{sub}$ and the complete readings $\mathcal{R}$.

$$\begin{aligned}\mathcal{L}(\mathcal{R}, \mathcal{R}_{sub}) &= \frac{RSS(\mathcal{M}(\mathcal{R}_{sub} \rightarrow \mathcal{R})) - RSS(\mathcal{M}(\mathcal{R}))}{RSS(\mathcal{M}(\mathcal{R}))} \times 100(\%) \\ &= \frac{\sum_{i=1}^{N}(R_k - \mathcal{M}(\mathcal{R}_{sub})|_{R_k})^2 - \sum_{i=1}^{N}(R_k - \mathcal{M}(\mathcal{R})|_{R_k})^2}{\sum_{i=1}^{N}(R_k - \mathcal{M}(\mathcal{R})|_{R_k})^2}\end{aligned} \tag{4}$$

where, $RSS(\mathcal{M}(\mathcal{R}_{sub} \rightarrow \mathcal{R}))$ means the RSS error for the approximation of the complete sequence $\mathcal{R}$ by using the model $\mathcal{M}(\mathcal{R}_{sub})$ that learnt from the sub sequence $\mathcal{R}_{sub}$.

Similar to reformulating the sensor placement problem in static WSN, there are two ways to represent the optimal sampling problem in OptiMoS: (1) *Given a limited sampling rate δ, find the best sampling set $\mathcal{R}_{sub}$ that has minimum information loss $\mathcal{L}(\mathcal{R}, \mathcal{R}_{sub})$*; and (2) *Given an acceptable information loss threshold ϵ between sampling sub-sequence $\mathcal{R}_{sub}$ and the complete sequence $\mathcal{R}$, find the best sampling points that has the minimal sampling rate*. They are formulated as the following two optimization problems respectively.

$$\underset{\mathcal{R}_{sub}}{argmin} \quad \mathcal{L}(\mathcal{R}, \mathcal{R}_{sub}) \quad s.t. \quad |\mathcal{R}_{sub}|/|\mathcal{R}| \leq \delta \tag{5}$$

$$\underset{\mathcal{R}_{sub}}{argmin} \quad |\mathcal{R}_{sub}|/|\mathcal{R}| \quad s.t. \quad \mathcal{L}(\mathcal{R}, \mathcal{R}_{sub}) \geq \epsilon \tag{6}$$

The sampling rate $|\mathcal{R}_{sub}|/|\mathcal{R}|$ is the sensing frequency, as a kind of mobile sensing cost. Thus, the optimal mobile sensing needs to balance the information loss (i.e., sensor coverage) with the sampling frequency (i.e., the energy cost). The previous two constraint optimization problems in Formula 5 and Formula 6 can be rewritten as an unconstrainted optimization in Formula 7, by using a balance coefficient λ.

$$\underset{\mathcal{R}_{sub}}{argmin} \ \mathcal{L}(\mathcal{R}, \mathcal{R}_{sub}) + \lambda|\mathcal{R}_{sub}| \tag{7}$$

IV. Segmentation of Mobile Sensing

This sections present various segmentation strategies in OptiMoS, from optimal segmentation by exhaustive search like dynamic programming, to top-down binary segmentation, to error-based heuristic and near-optimal segmentation. This is to solve the optimization problem in Formula 2.

A. Optimal Segmentation

The segmentation problem in Formula 2 has optimal solution using exhaustive search, where the algorithm is recursively searching the best point to divide i segments into i+1 segments. This can be computed by *dynamic programming* (DP) [4][11], with the complexity of $O(KN^2)$ where K is the segment number and N is the number of points in $\mathcal{R}$. With such quadratic complexity, DP is impractical for segmenting real-life large scale mobile sensing data. Nevertheless, we can apply DP to segment a short sequence using small $K(\leq 5)$, and evaluate other segmentation methods using the optimal modeling error from DP. The recursive function of using DP for segmenting the mobile sensor readings is summarized in Alg. 1. Thus, we can use $segmentDP(\mathcal{R}, 1, N, K)$ to solve Formula 2. To speed up the exhaustive search in DP, our experiment provides an extra condition of "bestRSSBound" (recording the best RSS error at each recursive step) to filter out some useless recursions of invoking $segmentDP$.

Algorithm 1: segmentDP $(\mathcal{R}, i, j, k)$

input : $\mathcal{R} = \{R_1, R_2, \cdots, R_N\}$ // mobile sensor readings
i, j // to make next segmentation in $\langle R_i, R_j \rangle$
k // the number of segments
output: $optimalRSS$ // model error by optimal segmentation

```
/* find the optimal segment */
if k = 1 and j > i then
    print i, j; // print optimal sub-segments
    return RSS(R, i, j); // compute model error RSS ⟨R_i, R_j⟩
/* impossible to find the optimal segment */
if j − i < k then
    return ∞;
/* recursive segmentation (from k to k-1) */
optimalRSS ← ∞; // initial the optimal RSS found so far
foreach id ∈ [i + 1, j − 1] do
    firstSegRSS ← RSS(R, i, id);
    restSegRSS ← segmentDP (R, id, j, k − 1);
    totalRSS ← firstSegRSS + restSegRSS;
    optimalRSS ← min{optimalRSS, totalRSS};
return optimalRSS;
```

It is worth noting that our objective is to find a segmentation that is not only the optimal for the training sequence, but also applicable to the similar sequences (as testing data). For example, OpenSense deploys environmental sensors (including CO_2, CO, temperature, humidity etc.) on the top of buses, for continuously monitoring air quality in the motion context. The segmentation result on one day of $Bus\text{-}line\text{-}1$ should be generally consistent with sensing data from others days on this same line. Therefore, the optimal segmentation from training day might not be the best for testing days. This is the overfitting issue which needs to be avoided in OptiMoS. In the experiment, we will show such experimental evidences.

B. Top-down Binary Segmentation

As the optimal segmentation by DP is impractical for real-life long sequence of mobile sensing data because of its high complexity, researchers proposed many greedy segmentation methods, such as the top-down binary split method [13]. The idea is to hierarchically split the sequence with maximum error into two sub-sequences, until the number of segments reaches K. We call this traditional top-down binary segmentation algorithm as $Binary$.

In $Binary$, the algorithm always choose a segment with the maximum model-based regression error (RSS) to make further segmentation, which may cause the division is only in one segment and its subsegments. As a result, the segmentation result could be totally unbalanced. This is similar to the worst case of a binary tree, where the tree is completely un-balanced and becomes a linked list. To overcome this problem, we design an extended algorithm of $Binary$, noted as $Binary^+$.

In $Binary^+$, we design a new RSS error measurement that has two types of penalties to prevent $Binary$ from always choosing the top RSS error: (1) how much error can be reduced after such segmentation; (2) what is the length size for the new segments (noted $length$), as shown in Formula 8, where α is the penalty coefficient. The first penalty is to evaluate the new segments in advance; and the second penalty is to avoid too short segments because of possible outliers. Alg. 2 provides the procedure of the $Binary^+$ algorithm.

$$\begin{aligned} newRSS &= RSS(\mathcal{M}(\mathcal{R}_{left})) + RSS(\mathcal{M}(\mathcal{R}_{right})) \\ \widehat{RSS} &= RSS(\mathcal{M}(\mathcal{R})) - newRSS + \alpha \times length \end{aligned} \tag{8}$$

Algorithm 2: segmentBinary$^+$ $(\mathcal{R}, K)$

input : $\mathcal{R} = \{R_1, R_2, \cdots, R_N\}$ // mobile sensor readings
K // the number of segments
output: $segOrderQueue$ // list of segments

```
/* initial the segment result */
segOrderQueue ← ∅;
/* insert the first segment to the sorted queue */
segOrderQueue.insert(R, 1, N);
foreach id ∈ [2, K] do
    /* retrieve & remove top error segment from the queue */
    topErrorSeg ← segOrderQueue.poll();
    /* divide the segment into two subsegments */
    S1 ← (R, topErrorSeg.begin, topErrorSeg.division);
    S2 ← (R, topErrorSeg.division, topErrorSeg.end);
    /* add the two new sub segments into two the sorted list*/
    calculate RSS^ for S1 and S2;
    segOrderQueue.insert(S1);
    segOrderQueue.insert(S2);
    segOrderQueue.resort(); // resort for next segmentation
return segList;
```

C. Heuristic Segmentation

The $Binary$ and $Binary^+$ methods focus on finding the maximum error segment (either RSS or $\widehat{RSS}$) to identify next segment to make division; but for the division point in the segment, they only apply the middle point, which is trivial. Therefore, we additionally design error-based greedy methods that use the model residual of each record to identify segment division. The residual is computed with the Formula 1. For such heuristic method, a simple greedy strategy is using the top error point as the division point for the segmentation. Recursively, we recompute the new models for new segments, and find the next top error point as the new division point, until reach K segments. This segmentation is called "$Heuristic$".

Similar to $Binary$ that always pick top error segment to make next segmentation, $Heuristic$ does the same strategy that always picks top error point as division point. Thus, we design an extended version called "$Heuristic^+$" that also uses the penalty function in Formula 8 to avoid that two division points are too close, i.e., the segment is too short. In addition, $Heuristic^+$ doesn't look for the largest error, but for the longest contiguous sequence of error exceeding a certain threshold (e.g. the error median) and then randomly choose one of its ending. This way it can isolate segments that have a bias in $\mathcal{M}(\mathcal{R})$.

The $Binary$ and $Binary^+$ methods study on finding the next segment and make division; whilst the $Heuristic$ and $Heuristic^+$ methods study on finding the next division points to make segmentation. The last segmentation method we propose in OptiMoS is B^+H^+ that combines $Binary^+$ and $Heuristic^+$. The idea is to consider both "the maximum error segment to build segmentation" but also "the maximum error point to make division". The combination is done as follows: the segment to divide is chosen by $Binary^+$ and then, inside this segment, $Heuristic^+$ is applied to find the segmentation point. This way we ensure that the segments have a better distribution, like in $Binary^+$, but also that the segmentation points is put in a region where the current model has its worst performance and thereby a certain improvement can be expected.

V. Sampling of Mobile Sensing

In this section, we study the second tier of OptiMoS, i.e., data sampling for each segment from the first tier. OptiMoS needs to balance the sensor coverage (i.e., minimizing modeling errors from mobile sensing samples) and the sensing cost (i.e., the size of samples). This is to solve the optimization problem in Formula 7, balancing the information loss $\mathcal{L}(\mathcal{R}, \mathcal{R}_{sub})$ and the sampling cost $|\mathcal{R}_{sub}|/|\mathcal{R}|$.

A. Optimal Sampling

The optimal sensor placement (or sampling) in an arbitrary sensing field is a NP hard problem [17][20][7]. For a simplified problem with a limited number of mobile sensing points (N), the optimal sampling at the sampling rate δ requires exhaustive search amongst all possible subsets of readings. This needs to consider all combination of δN points from the initial N points, which has the complexity of $O(N^{\delta N})$ and is still NP-complete. Therefore, we seek to near-optimal greedy solutions for optimal sampling in OptiMoS.

B. Distribution based Sampling

Intuitive greedy solutions for sampling the mobile sensor readings for each individual segment are using some statistical

distributions, e.g., $uniform$ and $normal$. In this paper, we evaluate the uniform sampling and the random sampling.

- *Uniform Sampling* – To uniformly select the δ percentage of mobile reading points in the segment, the algorithm selects the sensing point at each interval δN. We can apply such uniform sampling with $\delta N - 1$ times, and each time has different offset. The final accuracy of uniform sampling is the average of several trials.
- *Random Sampling* – In this method, we randomly select δN points from the segment, and make certain number of trials. Similar to uniform sampling, the final accuracy of random sampling is the average of several trials.

C. Entropy based Heuristic Sampling

In distribution based sampling, selecting sensing points only considers position distribution. Actually for the uniform sampling and the random sampling, each sensing point has the same chance to be selected. There is no bias in general.

To provide better sampling, we design error-based entropy sampling. Similar to the heuristic segmentation methods using modeling errors, we apply the residual $res(R_i)$ as the entropy for the selection of sensor readings. For example in Fig. 3, the segment initially has 25 sensor readings, i.e., the blue times symbol ("×") at the bottom. We plot the modeling residuals for each sensor reading in black line. By using these residuals, we can pick top 7 sensors as the red circles ("◯"), where three of them are mostly at the beginning of the segment and the four others are mostly at the end; therefore, the sampling looks not good and are bias to the absolute modeling errors.

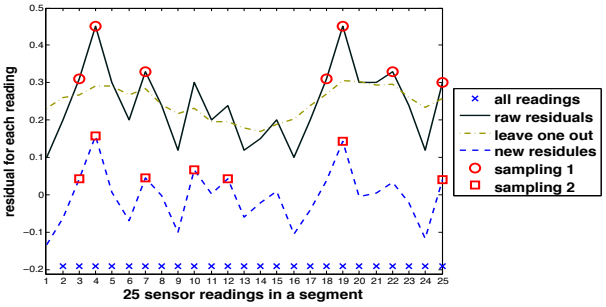

Fig. 3: Entropy based sensor readings sampling

Our improved solution in OptiMoS is to design an entropy value using a relative residual rather than the absolute value. The idea is using a *leave-one-out* basic model to approximate the residual and check how much is the gap between real residual and the approximated one (see Formula 9). To approximate the residual of point R_i, the *leave-one-out* model uses the residuals of nearby points (in window size $2w$, i.e. w for left and w for right) and builds simple interpolation (e.g., using basic mean, linear, Gaussian). In Fig. 3, the "leave-one-out" approximation is cumulated using the mean model with window size $2w$=4, and it is plotted in the gray dot-dashed line. Afterward, the gaps (i.e., $\hat{rés}(R_i)$) are calculated and plotted in the blue dashed line; and finally the new 7 sensor readings are selected as the new sampling in the red rectangle ("□"). We observe that the new sampling has more selections in the middle with a better distribution. In our experiment, we note such relative-error based entropy sampling as "$Entropy$".

$$\hat{rés}(R_i) = res(R_i) - leaveOneOutAppr(R_i, w) \quad (9)$$

D. Mutual Information based Heuristic Sampling

Both the absolute error res and the relative error $\hat{rés}$ in the entropy based mobile sampling are calculated in advanced. They are not recomputed during the sampling process. However, the entropy of sampling points actually are varying after each sampling step. Such information change can be modeled using the *mutual information* that can measure the mutual dependence of two random variables (in our case the candidate point to be selected and the points already been selected). Mutual information can reduce information dependency, therefore it has been largely used in many topics such as feature selection [24] and traditional static sensor placement [17]. Therefore, we additionally design mutual information based sampling method in OptiMoS.

Different from the previous entropy-based sampling method that directly selects points with top ranking relative residuals ($\hat{rés}$ in Formula 9), we redesign a loop procedure that can recursively recompute the new $\widetilde{rés}$ of the candidate sensor readings. Such new $\widetilde{rés}$ needs to remove the mutual information from sensor readings already selected (see Formula 10).

$$\widetilde{rés}(R_i; \mathcal{R}_{sub}) = \hat{rés}(R_i|\mathcal{R}_{sub}) - \hat{rés}(R_i) \quad (10)$$

where, $\mathcal{R}_{sub}$ is a set of sampling points already selected so far, R_i is a candidate reading for adding to $\mathcal{R}_{sub}$, $\hat{rés}(R_i|\mathcal{R}_{sub})$ is the relative residual computed only using the selected readings $\mathcal{R}_{sub}$, and $\hat{rés}(R_i)$ is the relative residual computed by all of the sensor readings $\mathcal{R}$ without sampling.

Algorithm 3: samplingMutualInfo $(\mathcal{R}, \delta)$

input : $\mathcal{R} = \{R_1, R_2, \cdots, R_N\}$ // mobile sensor readings
δ // the percentage of sampling
output: $\mathcal{R}_{sub}$ // sampling set, with size δN

```
/* initialization */
R_sub ← ∅; // initial empty sampling set
M ← int(δN); // the size of the final sampling set
/* get the first sample with pure entropy */
foreach R_i ∈ R do
    compute the entropy r̂és(R_i); // by Formula 9
firstSample ← argmax_{R_i} r̂és(R_i)
R_sub.add(firstSample);
/* get the following samples with mutual information */
while |R_sub| < M do
    foreach R_i ∈ R − R_sub do
        compute r̃és(R_i; R_sub); // by Formula 10
    nextSample ← argmax_{R_j} r̃és(R_j)
    R_sub.add(nextSample);
return R_sub;
```

Alg. 3 sketches the main procedure of using mutual information for getting near-optimal sampling. First, we apply relative error based entropy to pick up the first sampling point; then, we recursively compute the mutual information between candidate data reading and the selected data readings, and

choose the sensing point with the maximum new residual that removes the redundancy from existing sampling points. This procedure is stopped until the total sampling rate reaches δ.

VI. Experimental Evaluations

This section presents our experimental results of the two-tier optimal mobile sensing in OptiMoS. We evaluate OptiMos' different segmentation strategies and various sampling methods using real-life environmental data from mobile sensors.

A. Experimental Setup

To evaluate and compare the performances of these algorithms, we utilize the real-life mobile sensing data from the environmental monitoring project OpenSense [2]. OpenSense investigates informative air quality sensing in terms of deploying sensors on top of several public transport vehicles such as buses and trams in Switzerland, to achieve a large coverage of the city area. The longterm objective is to raise additional community sensing in the environmental sensing campaign, using enhanced smartphone/pocket sensors [8] and private vehicles [16].

In OpenSense, public transport vehicles are deployed with location sensor (GPS) and several environmental measurement sensors including $temperature$, $humidity$, CO, CO_2, NO, NO_2, $fine\ particles$ etc. In current deployment, the measurement frequency of these mobile sensors are fixed at $\frac{1}{60}$Hz, i.e., one sensing record R_i per minute. Fig. 4 shows the distribution of CO_2 measurements from several bus lines in the Lausanne city area during two weeks, where the points are the mobile sensing locations and the colors indicates the level of CO_2 value. This is our early experimental Web interface for querying and visualizing such mobile sensing data from the OpenSense project. This Web implementation applies OpenStreetMap [3] as the embedded map interface.

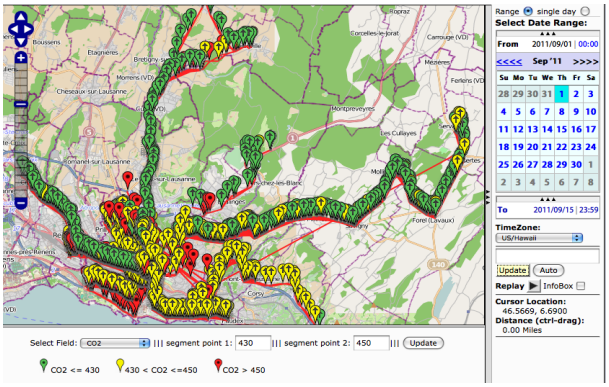

Fig. 4: Distribution of CO_2 measurements from moving buses

To take a concrete look at various mobile sensor measurements, Fig. 5 plots one day mobile sensing of several sensors deployed on a bus line in Lausanne; the environmental sensors include $temperature$, $humidity$, CO, NO_2, CO_2. From Fig. 5, we observe that CO_2 has large variance compared to other environmental sensors readings (i.e., CO and NO_2). Due to its large variance for data prediction, modeling on CO_2 could be more interesting and challenging; therefore, in our experimental study, this paper focuses on investigating the optimal mobile sensing of the CO_2 measurement.

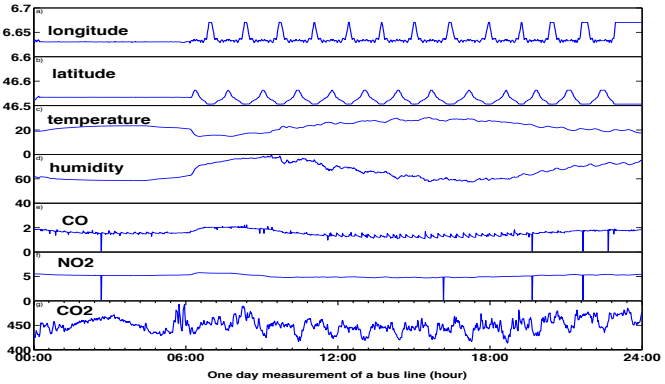

Fig. 5: One day mobile sensing with various measurements

The mobile sensing objective in OpenSense is to find an optimal sampling strategy to replace current OpenSense's uniform sampling. This obviously will reduce sensing cost and save energy. The longterm goal is designing a global optimal sensing for multiple bus lines in the city area, as previously shown in Fig. 1 with five bus lines. In this paper, we target an optimal sensing solution for each individual bus line.

B. Data Model Implementation

There is no model restriction in our two-tier optimal sensing framework. As sketched in Fig. 2 and discussed in Section III, OptiMoS supports all kinds of models such as linear model, polynomial model, SVM regression, time series ARIMA model etc. At the moment, our experiments test two types of models, i.e., the basic linear model and the SVM regression. Linear model is simple and efficient to compute; whilst SVM can achieve high accuracy but requires more computing cost. For SVM implementation, we use the LibSVM package [6], which is largely applied in the literature because of its high computing efficiency. All the input data (e.g., CO_2 values) were also linearly scaled to $[0, 1]$ for normalization.

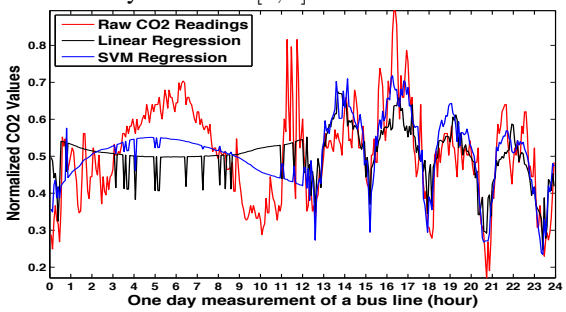

Fig. 6: Regression for the whole segment

Fig. 6 shows the regression results using the two models (i.e., $Linear\ Regression$ and $SVM\ Regression$) compared to the raw complete sensor readings at $\frac{1}{60}$Hz; this is a 24-hour mobile sensing without any segmentation. By visual comparison in Fig. 6, the approximation by SVM regression is

close to the ground truth values, which is better than the linear model; more precisely, the total model error $RSS(\mathcal{M}(\mathcal{R}))$ using SVM is 0.0802, whilst $RSS(\mathcal{M}(\mathcal{R}))$ from the linear model is 0.1004. As shown in Fig. 6, the approximation is not very good (particularly at the duration of 0am-10am), and we will show better approximation results by using segment-based regression in the next subsection.

C. Segmentation Results

To evaluate various segmentation methods proposed in Section IV and make comparison, we define a metric for quantifying the model error reduction by segmentation (from the initial non-segment sensor readings $\mathcal{R}$ to the K segments $\mathcal{R}_i$), i.e., the ratio of modeling errors called "RSS_Ratio".

$$RSS\_Ratio = \frac{\sum_{i=1}^{K}(RSS(\mathcal{R}_i))}{RSS(\mathcal{R})} \times 100(\%) \qquad (11)$$

Fig. 7 presents the RSS_Ratio achieved using SVM regression on one-day mobile sensing as the training data. The six segmentation methods (i.e., $Binary$, $Binary^+$, $Heuristic$, $Heuristic^+$, B^+H^+, $Optimal$) are experimented with different segment numbers, from 2 to 10. Clearly, with more segments (i.e., larger K), the error ratio can decrease more. Initially, such error decrease is significant at small K, but later it becomes more stable when K becomes larger.

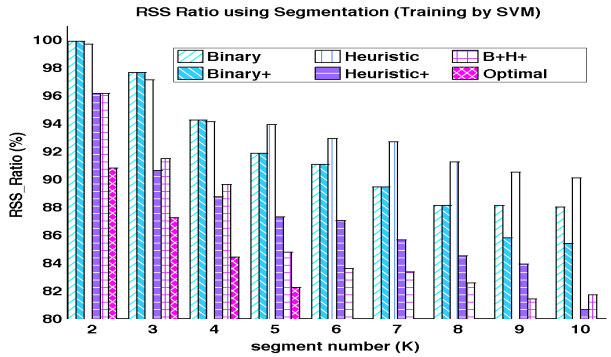

Fig. 7: Training on Day-1 by SVM

For individual segmentation methods on the training RSS_Ratio errors, we observed that the $Heuristic^+$ (or B^+H^+) method is better than other segmentation methods. We additionally compare them with the optimal solution ($Optimal$) using dynamic programming when the segment number is small (i.e., $K \leq 5$). We omit such optimal solution for larger K, as the computation time is indeed expensive for dynamic programming. The RSS_Ratio achieved by the $Heuristic^+$ (or B^+H^+) method is closer to the optimal solution compared to other methods. Therefore, error-based heuristic method is good for model-driven segmentation.

To further evaluate the segmentation results learned from one day training data, we test them on mobile sensing data from other days at the same bus line. This test can evaluate the robustness of our segmentation results. Fig. 8 shows such testing errors of RSS_Ratio. We observe that $Heuristic^+$ is clearly better than other methods, and even better than the $Optimal$ for most cases ($K = 3, 4, 5$). This is the over-fitting problem, i.e., the optimal segmentation for training data is not necessarily the best for the testing data.

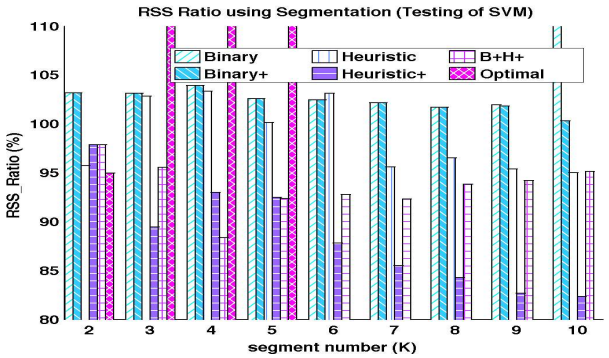

Fig. 8: Testing on Day-2 using SVM-inferred segments

Fig. 7 and Fig. 8 presented the segmentation results using the SVM regression model. For linear regression, we observe similar trends amongst different segmentation methods. Additionally, the optimal segmentation using linear model in the training data works even worse on the testing data. Linear model has more prominent over-fitting problems compared to the SVM modeling. Therefore, in future, we need to further study more robust optimal segmentation that works not only for training data but also for the test ones.

In contrast to Fig. 6 about regression without segmentation, Fig. 9 shows regression with segmentation (learnt by $Heuristic^+$ when K=5). We observe better regression results by using segmentation, particularly at the duration of 0am-10am. In terms of the concrete modeling errors, $RSS(\mathcal{M}(\mathcal{R}))$ using SVM regression is 0.070 and $RSS(\mathcal{M}(\mathcal{R}))$ by linear model is 0.081; the two models respectively have model error decrease 12.7% and 19.3% compared to their modeling errors without segmentation shown in Fig. 6. Therefore, segmentation can gain more error reduction for simpler model (e.g., linear) compared to an advanced one (e.g., SVM).

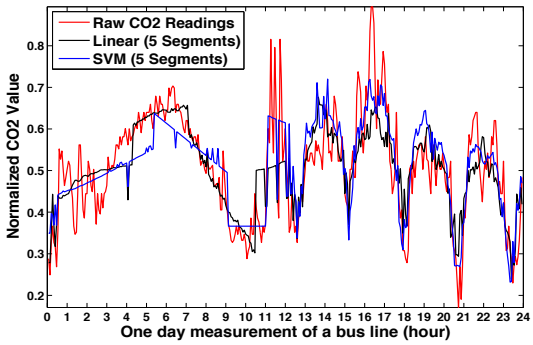

Fig. 9: Regression in segments ($Heuristic^+$, $K = 5$)

D. Sampling Results

To evaluate different sampling methods that we presented in Section V, we can use the metric of information loss (see Formula 4). However, for a very long sequence of mobile sensor readings without segmentation, the information loss can

be negative, as the initial regression on the complete sequence can be worse than a modeling with a good subsequence of sampling points. Therefore, we modify the definition of the exact information loss ($\mathcal{L}$) and apply $\widehat{\mathcal{L}}(\mathcal{R}, \mathcal{R}_{sub})$ (i.e., the direct RSS ratio) to evaluate different sampling methods (see the following Formula 12).

$$\widehat{\mathcal{L}}(\mathcal{R}, \mathcal{R}_{sub}) = RSS\_Ratio = \frac{RSS(\mathcal{M}(\mathcal{R}_{sub} \rightarrow \mathcal{R}))}{RSS(\mathcal{M}(\mathcal{R}))} \times 100(\%) = \frac{\sum_{i=1}^{N}(R_k - \mathcal{M}(\mathcal{R}_{sub})|_{R_k})^2}{\sum_{i=1}^{N}(R_k - \mathcal{M}(\mathcal{R})|_{R_k})^2} \quad (12)$$

Fig. 10 presents the experimental results of using four sampling methods, i.e., $Uniform$, $Random$, $Entropy$ and $Mutual\_Information$, when using various sampling rates (i.e., $\delta = 1/2, 1/3, 1/4, 1/5$). In the plot, the RSS_Ratio from the $Uniform$ and $Random$ sampling methods are the average over 10 runs. It is interesting to notice that the uniform sampling on the average has better performance than the random sampling. Both $Entropy$ and $Mutual\_Information$ sampling methods have quite good results. There is almost no error increase compared to the original data sequence when using $Entropy$ and $Mutual\_Information$ based samplings at high sampling rate ($\delta \geq 1/4$). Their RSS_Ratio are almost 100%. At sampling rate $\delta = 1/5$, we clearly see that $Mutual\_Information$ performs better than $Entropy$.

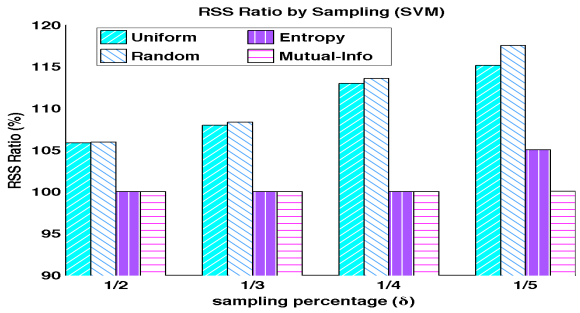

Fig. 10: RSS Ratio by different sampling methods (SVM)

To further study the performance of $Entropy$ and $Mutual\_Information$ methods at smaller sampling rate ($\delta < 1/5$), Fig. 11 shows the results of sampling using linear model. We observe the similar trend with SVM model in Fig. 10 at $\delta \geq 1/5$, i.e. $Mutual\_Information$ is the best for all cases. It is worth noting that $Entropy$ has better performance at high sampling rate ($\delta \geq 1/5$), almost the same as $Mutual\_Information$ when ($\delta \geq 1/4$), but it goes to significantly worse when the sampling rate is too small ($\delta < 1/5$). This is because $Entropy$ sampling is always choosing the top-ranking error points, without consideration of current selected points. This can work well for large sampling rate, but for small sampling rate, the data points will be very bias and have poor performance.

We already see with sampling rate $\delta \geq 1/4$ in Fig. 11 and Fig. 10, both $Entropy$ and $Mutual\_Information$ sampling methods can guarantee almost 100% RSS ratio, i.e., minimal informational loss using sampling data compared to using the complete data. To further study how much sampling is

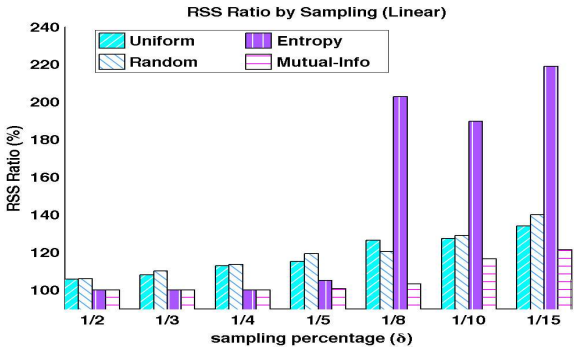

Fig. 11: RSS Ratio by different sampling methods (Linear)

requested for achieving such minimal informational loss, we study the convergence of RSS_Ratio w.r.t. the sampling rate. Fig. 12 shows the RSS_Ratio convergence for sampling the one-day mobile sensing measurement (totally 1440 points in 24 hours with one record per minute). We observe that both linear and SVM regression can achieve almost 100% RSS_ratio from 128 sampling points, i.e., the sampling rate at $\delta = 128/1440 \simeq 1/11$. In such case, the sampling rate at $1/11$ can guarantee almost zero information loss modeling (around 110% RSS ratio), which is very effective sampling.

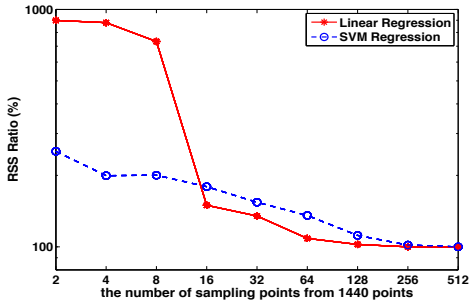

Fig. 12: RSS Ratio convergence using Mutual-Information sampling

E. Near-Optimal Combination

The final experiment is to study the exhaustive strategies of mobile sensing in OptiMoS, i.e., combining different segmentation and sampling methods. Due to the limitation of space, we do not show all possible combinations. In addition, for segmentation, previous experimental results already showed the good performance of $Heuristic^+$ (B^+H^+ is not always better than $Heuristic^+$); for sampling, the performance of $Entropy$ based sampling is as good as $Mutual\_Information$ when sampling rate is reasonably high ($\delta \geq 1/4$). Therefore, Fig. 13 shows the 4×3 mobile sensing strategies in OptiMoS, i.e., combining four segmentation methods (i.e., $Binary$, $Binary^+$, $Heuristic$, $Heuristic^+$) with three sampling methods ($Uniform$, $Random$, $Entropy$) using SVM regression.

For each combination in Fig. 13, the plot shows the convergence of RSS_Ratio w.r.t. the segment number (K) at different sampling rate (δ). All plots present the decrease trend of RSS_Ratio with more segments (from 1 to 10), and with larger sampling ratio (from $1/4$ to $1/1$). The $Entropy$ sampling with heuristic segmentation (both $Heuristic$ and

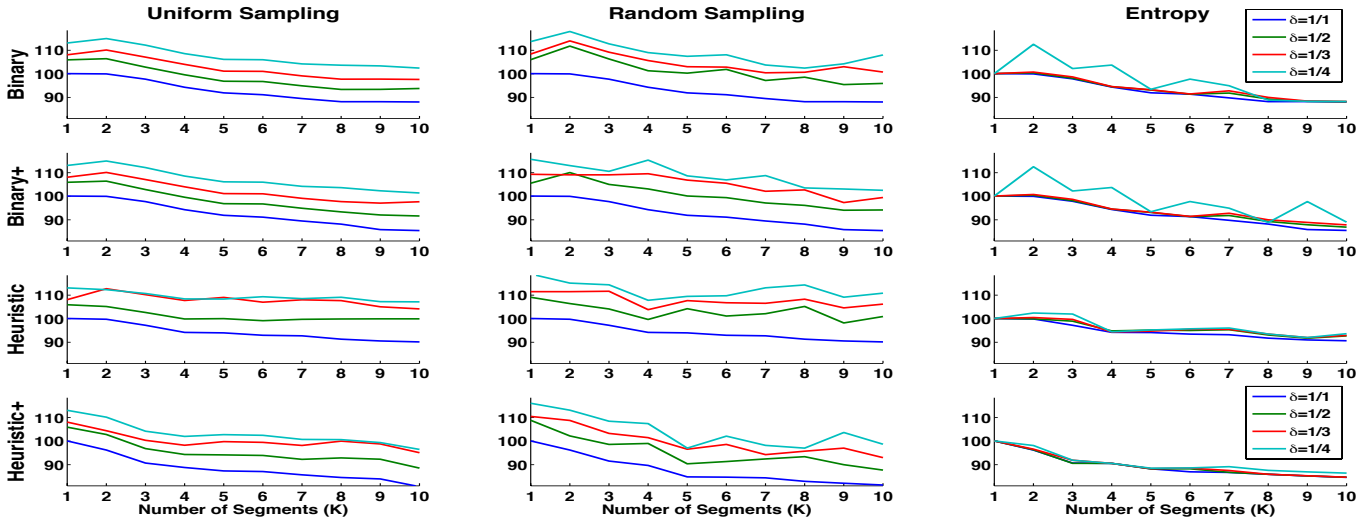

Fig. 13: RSS Ratio at different combinations of segmentation and sampling (SVM)

$Heuristic^+$) show good convergence when segment number $K \leq 3$. The combination of $Entropy$ and $Heuristic^+$ keeps the best RSS_Ratio convergence, even δ is small (say $1/4$).

VII. Conclusion

In this paper, we proposed a novel two-tier framework, namely OptiMoS, that enables an optimal mobile sensing strategy. As far as we know, this is the first paper that studies the problem of optimal sensing for mobile sensors. We studied both segmentation and sampling of mobile sensor readings, and designed several methods for segmentation (e.g., $Optimal$, $Binary$, $Binary^+$, $Heuristic$, $Heuristic^+$, B^+H^+) and sampling (e.g., $Uniform$, $Random$, $Entropy$, $Mutual\_Information$). We analyzed real-life environmental monitoring sensors deployed on moving buses, built exhaustive empirical studies to validate this optimal mobile sensing strategy, and showed its good performance.

Our future work is to further extend OptiMoS for global optimal sensing on multiple bus lines in a large city area. In addition, we will study the co-sensing strategy amongst various environmental sensors such as CO_2, CO and NO.

References

[1] Floating Sensor Network. http://lagrange.ce.berkeley.edu/fsn/.

[2] OpenSense. http://opensense.epfl.ch.

[3] OpenStreetMap. http://www.openstreetmap.org/.

[4] R. Bellman. On the Approximation of Curves by Line Segments Using Dynamic Programming. *Communications of the ACM*, 4(6):284, 1961.

[5] M. Buchin, A. Driemel, M. J. van Kreveld, and V. Sacristan. Segmenting Trajectories: a Framework and Algorithms Using Spatiotemporal Criteria. *J. Spatial Information Science*, 3(1):33–63, 2011.

[6] C.-C. Chang and C.-J. Lin. Libsvm: a Library for Support Vector Machines. *ACM TIST*, 2(3):27, 2011.

[7] S. Dhillon and K. Chakrabarty. Sensor Placement for Effective Coverage and Surveillance in Distributed Sensor Networks. In *WCNC*, pages 1609–1614, 2003.

[8] P. Dutta, P. Aoki, N. Kumar, A. Mainwaring, C. Myers, W. Willett, and A. Woodruff. Common Sense: Participatory Urban Sensing Using a Network of Handheld Air Quality Monitors. In *SenSys*, pages 349–350, 2009.

[9] M. Faulkner, M. Olson, R. Chandy, J. Krause, K. Chandy, and A. Krause. The Next Big One: Detecting Earthquakes and Other Rare Events from Community-Based Sensors. In *IPSN*, pages 13–24, 2011.

[10] E. Fuchs, T. Gruber, J. Nitschke, and B. Sick. Online Segmentation of Time Series Based on Polynomial Least-Squares Approximations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(12):2232–2245, 2010.

[11] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmäki, and H. Toivonen. Time Series Segmentation for Context Recognition in Mobile Devices. In *ICDM*, pages 203–210, 2001.

[12] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange. SensorScope: Application-Specific Sensor Network for Environmental Monitoring. *TOSN*, 6(2):17:1–17:32, 2010.

[13] E. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting Time Series: A Survey and Novel Approach. In *Data mining in Time Series Databases. Published by World Scientific*, pages 1–22, 2004.

[14] E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani. An Online Algorithm for Segmenting Time Series. In *ICDM*, pages 289–296, 2001.

[15] A. Krause and C. Guestrin. Optimizing Sensing: From Water to the Web. *IEEE Computer*, 42(8):38–45, 2009.

[16] A. Krause, E. Horvitz, A. Kansal, and F. Zhao. Toward Community Sensing. In *IPSN*, pages 481–492, 2008.

[17] A. Krause, A. Singh, and C. Guestrin. Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

[18] D. T. Lee and A. K. Lin. Computational Complexity of Art Gallery Problems. *IEEE Trans. on Information Theory*, 32(2):276–282, 1986.

[19] D. Lemire. A Better Alternative to Piecewise Linear Time Series Segmentation. In *SDM*, pages 545–550, 2007.

[20] F. Lin and P. Chiu. A Near-Optimal Sensor Placement Algorithm to Achieve Complete Coverage-Discrimination in Sensor Networks. *IEEE Communications Letters*, 9(1):43 – 45, 2005.

[21] S. Lin, B. Arai, D. Gunopulos, and G. Das. Region Sampling: Continuous Adaptive Sampling on Sensor Networks. In *ICDE*, pages 794–803, 2008.

[22] X. Liu, Z. Lin, and H. Wang. Novel Online Methods for Time Series Segmentation. *IEEE Trans. Knowl. Data Eng.*, 20(12):1616–1626, 2008.

[23] T. Palpanas, M. Vlachos, E. J. Keogh, D. Gunopulos, and W. Truppel. Online Amnesic Approximation of Streaming Time Series. In *ICDE*, pages 339–349, 2004.

[24] H. Peng, F. Long, and C. Ding. Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. *IEEE PAMI*, 27(8):1226 –1238, aug. 2005.

[25] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu. Ear-Phone: an End-To-End Participatory Urban Noise Mapping System. In *IPSN*, pages 105–116, 2010.

[26] G. Tolle, J. Polastre, R. Szewczyk, D. E. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A Macroscope in the Redwoods. In *SenSys*, pages 51–63, 2005.

[27] Z. Yan, C. Parent, S. Spaccapietra, and D. Chakraborty. A Hybrid Model and Computing Platform for Spatio-semantic Trajectories. In *ESWC*, pages 60–75, 2010.