POOLVIEW: TOWARDS A PEOPLE CENTRIC SENSING WORLD

BY

RAGHU KIRAN GANTI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Doctoral Committee:

Associate Professor Tarek Abdelzaher, Chair
Assistant Professor Nikita Borisov
Professor Klara Nahrstedt
Professor John Stankovic, University of Virginia
Professor Kenneth Watkin

Abstract

The availability of a wide variety of networked sensing devices in the form of everyday devices such as smartphones, music players, smart residential power meters, sensor embedded gaming systems, and in-vehicle sensing devices will result in the evolution of an embedded Internet. In this scenario, the main role of the Internet and its applications will shift gradually from offering a mere communication medium between end-points to offering *information distillation* services bridging the gap between the varied data feeds from the sensing devices and human decision needs. In this thesis, we take a step towards the development of an architecture and a data analysis toolset for realizing the above vision of the future Internet. In particular, we focus on a category of sensing, called people centric sensing, where the sensing devices are owned by individuals. We present various novel generic data analysis tools that are necessary to enable people centric sensing applications. We take a systems approach and exemplify these tools by developing and implementing prototypes of several people centric sensing applications. We also provide extensive data collection and evaluation for each of the exemplified applications, which show the utility of our architecture.

In memory of my dear father, Annaji Rao, who will always be my guiding light

To my mother, Venkataramana,

my brother, Vamsi,

and my dear wife, Avanija.

Acknowledgements

My Ph.D. journey was an exciting one, with many ups and downs, transitions across two schools and to a certain extent bordering being dramatic. I started this journey at the University of Virginia, Charlottesville and am ending it at the University of Illinois, Urbana-Champaign. During the course of this journey, first and foremost, I would like to thank my family members, my mother, father, brother, and wife for their constant support and faith. Without their encouragement, it would have not been possible to achieve anything at all. The end of this journey was on a very sad note, my father passed away before my final defense. I would like to dedicate this thesis to his memory and add that he will live with me in spirit always and his ideals will guide me through the rest of my life, as always.

This thesis would not have been even remotely possible without my adviser and teacher, Prof. Tarek Abdelzaher. In my home country, we have a saying, a part of which translates to "*The teacher is the supreme spirit - I offer my salutations to this Teacher*". In my case, it is literally true. I am deeply indebted to my adviser who has been a constant source of encouragement and support. He never gave up on me, in spite of my several shortcomings and mistakes. I learned a whole lot from him in every aspect, including research and personal. There is not enough space in this thesis nor words that would describe the countless things that I have learnt from Tarek. Perhaps, my wish of remaining a student of Tarek's forever would capture these feelings of mine.

I would like to thank my committee members, Prof. Jack Stankovic, Prof. Kenneth Watkin, Prof. Klara Nahrstedt, and Prof. Nikita Borisov for their inputs, which helped improve this thesis. Their periodic feedback on the work done in this thesis and also their inputs during my job hunt phase proved invaluable. Thanks are due to Prof. Haiyun Luo and Prof. Indranil Gupta, who inspired new research ideas during the early stages of my Ph.D. I would also like to thank Prof. Geneva Belford and Prof. Steve LaValle, who were immensely helpful during my transition to UIUC.

I am grateful to the various agencies that provided funding for the work done in this thesis, including NSF and Microsoft research. Special thanks are due to the Siebel foundation and Tom Siebel for the generous Siebel scholar fellowship awarded during my last year of stay at UIUC.

I would also like to thank the various inputs provided by the anonymous reviewers and shepherds for my papers. It helped improve the work done in this thesis significantly.

Thanks are due to Soundararajan Srinivasan and Aca Gacic, who as mentors during my internship at Robert Bosch research center were very helpful. I am also grateful for the permission from Robert Bosch to use the work done as part of the internship in my thesis.

I am also indebted to my cousin and his wife, Pushpakumar Ganti and Sunitha Ganti, who were of great help when I arrived in the United States for the first time and also when I moved to UIUC. Also, I would like to mention their two extremely cute daughters, Sree Pranathi and Sree Hamsini.

Friends are an integral part of life, both at work and outside. All my friends helped me in a multitude of ways during my stay in Urbana-Champaign and Charlottesville. At the University of Virginia, my friends Sharanya Eswaran, Ankit Malhotra, and Yogesh Tiwary were a great source of entertainment and support during my stay. Thao Doan and Anthony Wood were of immense help for my experiments with the *smart jacket*. Although the dreary corn field studded landscape of Illinois was a significant aesthetic setback from the lush green Blue Ridge mountains of Virginia, my research took new directions at the University of Illinois, Urbana-Champaign. The first inspiration was from Praveen Jayachandran (a.k.a. JP), with whom I worked closely for over a year. He taught me the importance of theory and analysis during these transitional years and also became a great friend. Yet another motivational person was Pradeep Ramachandran, who was a nonstop source of great entertainment and above all someone who taught me the importance of being positive. During my latter years, I made a whole lot of friends, Aravind Alwan, Harini Barath, Milu Cherian, Gowri Kannan, Vinay Natarajan, Chandu Parimi, Kaushik Ragunathan, Swetha Ravi, Vivek Srikumar, Vikhram Swaminathan, Vidisha Vachharajani, and Preeti Warty. Special mention is due to Aari Parimi (at the time of writing this thesis, a one year old toddler), who had an equal role as my other friends. We formed a great group and they are invaluable friends, we cooked together, organized several charity programs, ate out, and explored various places too! They were extremely important in maintaining a balance between my work and life.

It is hard to pen down all the names of the people who have contributed to my thesis in one way or the other. And, considering that it has been seven years since I started my pursuit towards a doctoral degree, I would have definitely missed mentioning a whole lot of them. I would like to thank each and everyone of you who have helped me achieve the work done in this thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The integration of sensing and embedded computing devices at the edge of the Internet will result in the evolution of an *embedded* Internet. This evolution is fueled by the pervasive availability of networked everyday sensing and computing devices such as smartphones (iPhone), music players (iPod + Nike), smart residential (wireless) power meters, sensor embedded gaming systems (e.g. Wii), and in-vehicle sensing (GPS, OBD-II) devices. These trends have been identified and shared by Internet pioneers such as David Clark, the network's former chief architect, in his motivational keynote speech at NSF's FIND (Future Internet Design) initiative [28]. In this scenario, the main role of the Internet and its applications will shift gradually from offering a mere communication medium between end-points to offering *information distillation* services bridging the gap between the varied data feeds from the sensing devices and human decision needs. The success of Google, built around the mission of organizing the world's information and making it universally accessible and useful, already attests to the increasing use of the Internet as an information source. Some of the important research challenges in enabling this future Internet are the identification and organization of various tools for the extraction, processing, and analysis of information from sensory data feeds. This thesis is a step towards the development of an architecture and a data analysis toolset for realizing the above vision of the future Internet. In particular, we focus on the tools required for enabling *people centric sensing*, where the edge sensors and embedded computing devices are owned by individuals (e.g. smartphones, GPS devices, cars). We can further categorize people-centric sensing into two types: (i) *personal sensing* and (ii) *community sensing*. In personal sensing, individuals collect sensor data and consume it for their own needs. For example, individuals keep track of the exercises they do (using a smartphone) and monitor their health. On the other hand, in community sensing, individuals collect sensor data and contribute towards a common goal, such as the computation of community statistics or mapping of global phenomenon. For example, individuals measure the CO_2 content of the air from their

daily commutes and share the data (with an *aggregation* server) to build a map of the carbon footprint in a given area, which can be used to design strategies for improving air quality.

A key challenge that this thesis addresses is the identification, development, and analysis of a generic set of tools required for enabling the efficient development of people centric sensing applications. These tools are organized in the form of an architecture, *PoolView*, with standard APIs. PoolView is built as a collection of these generic data analysis tools, which collectively form an application layer service of the Internet. The PoolView architecture and toolset are exemplified through various real world deployments, which focus on improving future healthcare and decreasing the consumption of non-renewable sources of energy on a global scale (and thus reduce the carbon footprint).

PoolView adopts a client-server approach, where clients are individuals who collect sensor data from everyday devices such as smartphones, iPods, GPS, and cars and servers aggregate the sensor data contributed by the individuals. With this in mind, we can be broadly classify the components of PoolView into *personal* (client) and *community* (server) categories. The personal or client side components include (i) Data formatters (ii) Data storage/retrieval, (iii) Activity identification, and (iv) Privacy firewall for sensor data sharing. The community or server side components comprise (i) Data storage/retrieval, (ii) Community statistics computation from shared sensor data, (iii) Community data modeling from shared sensor data, and (iv) Map based application support tools. In what follows, we will discuss three major tools, personal monitoring, privacy preservation and community statistics computation (the privacy preservation and community statistics tools are related to each other, which we will discuss below), and community data modeling, as they pose interesting research challenges. We will also discuss the choice of these tools and the corresponding applications that exemplifies each of these tools.

1.1 Tool 1: Personal Monitoring

The first piece of my work focused on the distillation of sensor data feeds from wearable devices for the purpose of personal consumption. Of particular interest is a category of services termed as *personal monitoring* services. Personal monitoring services are software services that enable the monitoring of daily human activities through the collection and analysis of sensor data from devices that interact with their user on a daily basis. These services will find use cases in several application domains, such as healthcare, social

networking, entertainment, and personal record keeping. An example healthcare application is one in which a patient needs to be monitored at home for a prolonged period of time by a care-giver. An example social networking application is where individuals share their activity information on social networking sites such as Facebook. A personal record keeping application example is where individuals keep track of their exercise records over the course of a month. The main challenge to enable such services is to develop a general framework that can identify the activities performed by the user. Although, human activity identification has been extensively addressed in the past [41, 91, 92, 96], most of them have used specialized devices or lab based environments to achieve human activity monitoring. Recently [47, 76, 46], everyday sensing devices have been used to achieve activity identification. The use of specialized devices allows for tuning of the hardware to achieve application specific optimization, whereas devices such as smartphones are not optimized for identification of activities. Such devices are equipped with possibly low quality sensors or are constrained by the device's primary functionality. Further, an important goal is that of transparency to the user, where the device monitors activities of the individual with minimal intervention to their daily lives. We address these challenges by developing a novel framework for the identification and monitoring of human activities using everyday sensing devices. Towards this end, we demonstrate this framework using two real-world prototypes, one uses MicaZ motes embedded in a jacket and the second uses smartphones[1].

1.2 Tool 2: Privacy Preservation and Community Statistics

The second aspect of my work focuses on community sensing, where individuals collect sensor data and share it among themselves to map common phenomena or compute community statistics. Earlier community sensing applications [22, 35, 61, 94] have focused on data collection and its analysis. Another important aspect of these applications is the potentially sensitive nature of sensor data being shared. For example, GPS sensor readings can be used to infer private information about the individual, such as the routes taken by the individual during their daily commutes, their home location, their work location, and so on. On the other hand, these GPS sensor readings (from daily commutes) shared within a larger community can be utilized to map the traffic patterns in a given city by computing various statistics related to traffic scenarios. Hence, an important aspect that needs to be addressed to enable such community applications is the data privacy

[1]Data collection from all experiments involving human subjects were approved by UIUC's IRB (#06703 and #10092)

of an individual. The importance of privacy in Internet based applications has been emphasized in [51], a classic paper that motivated future privacy research. We are interested in a *grassroots* solution that enables the sharing of data (in a privacy preserving manner) in the absence of a trust hierarchy. Similar to the Web, where any individual can create a webpage and share information, we wish to empower the common person to be able to create new applications (that utilize sensor data collected by other individuals within a community). Existing privacy approaches such as anonymization [100] are not useful in this scenario. For example, anonymized GPS (location) sensor measurements can be used to infer the frequently visited locations of the individual and derive their personal details (in many cases). Secure multiparty computation approaches [52] on the other hand are compute intensive and are not scalable (require the generation and maintenance of multiple keys). We adopt the approach of data perturbation, the addition of noise to sensor data before sharing it with the community to achieve privacy. The challenge in this scenario is to add noise in such a manner that the privacy of the individual is preserved, but at the same time, it is possible to compute the statistics of interest with a high accuracy. Data perturbation approaches [5, 4] have been repeatedly shown [59, 66, 85] to be unable to preserve privacy under certain conditions (when the data being shared are correlated). Hence, the main challenge arises due to the time series nature of most sensor data, which results in the sensor measurements being correlated with each other.

We develop a novel technique (this thesis only claims a partial contribution towards the development of this privacy preservation algorithm) that allows for the sharing of time series sensor data in a privacy preserving manner within a community and also the reconstruction of accurate community statistics. Specifically, this thesis applies the above privacy preserving technique to two applications, “Traffic Analyzer” and “Weight Watchers”. In Traffic Analyzer, individuals record GPS sensor readings from their daily commutes and share the corresponding perturbed sensor measurements with an aggregation server. These perturbed sensor measurements are then used by the aggregation server to correctly compute various traffic measures within a given community (in this case, Urbana-Champaign). Individuals who use the Weight Watchers application share their perturbed weight measurements with an aggregation server, which computes the statistics of the community (e.g. average weight of individuals in the community, number of people above a certain weight).

1.3 Tool 3: Community Data Modeling

Typical community sensing applications [35, 61] have focused on the aspects of development of data collection and its analysis. But, several of these applications require large amounts of data to compute or map the phenomena of interest. For example, in order to obtain a complete traffic map (CarTel [61] deployments are small scale and compute traffic statistics from only the data collected) of a large city (e.g. Chicago, Seattle), individuals need to contribute a large number of traffic measurements within the city (in order to obtain complete coverage). Typically, initial deployments will be sparse as the usefulness of the application is realized. An important challenge in this scenario is to be able to generalize from relatively sparse measurements of high-dimensional spaces to model the phenomena of interest. This is complicated by the fact that such phenomena are complex and trivial modeling techniques (e.g. linear regression) will fail to capture the entire phenomena. In this thesis, we will illustrate a solution methodology for the generalization problem using a green navigation application, GreenGPS. GreenGPS is a GPS-based navigation service that gives drivers the most fuel-efficient route for their vehicle as opposed to the shortest or fastest route. In order to obtain the fuel-efficient routes, GreenGPS maps out the fuel consumption of any car on any given street. This is achieved through the combination of fuel consumption data collected by a few individuals from their vehicles and a generalization framework that predicts the fuel consumption of an arbitrary car on an arbitrary street.

1.4 Contributions

The generic contributions of this thesis can be categorized as follows:

- This thesis proposes PoolView, the first architecture and toolset that enables easy and efficient development and deployment of people centric sensing applications.
- We also develop the first activity identification framework that classifies progressively more complex activities using multiple sensor inputs from everyday devices such as smartphones.
- This thesis claims partial contributions towards the development of a privacy preserving technique that enables the sharing of time-series sensor data such that the privacy of an individual is preserved,

while enabling the computation of community statistics accurately.

- This thesis claims partial contributions in the development of a generalization methodology that models large scale phenomena from relatively sparse measurements of high-dimensional spaces.

Further, the contributions of this thesis which are application specific are as follows:

- This thesis develops the first traffic analyzer application that enables the computation of traffic related statistics accurately when individuals share perturbed GPS sensor readings. The perturbation of these sensor measurements is applied in such a way that the privacy of an individual is preserved.

- This thesis develops the first green navigation application, that provides drivers with the fuel efficient route (as opposed to fastest or shortest routes) between arbitrary points in a given city.

1.5 Thesis Outline

The rest of this thesis is organized as follows. Chapter 2 will present the architecture of PoolView and the various interfaces between the tools. Chapter 3 presents the personal monitoring tool and the corresponding real world deployments, smart jacket and smartphone. Chapter 4 discusses tools that preserve privacy while sharing sensor data and compute accurate community statistics, which are exemplified by the Traffic Analyzer and Weight Watchers applications. Chapter 5 illustrates the modeling tools required for community sensing applications, which is utilized by the GreenGPS application. We discuss related work in Chapter 6. We conclude with lessons learned, impact of this thesis, and directions for future work in Chapter 7.

Chapter 2

PoolView Architecture

In this chapter, we will introduce the architecture of PoolView, its components, APIs, and their functionalities. Before describing the architecture, we will motivate the need for one and the challenges in its development. The wide variety of sensing and computing devices (e.g. smartphones, in-vehicle GPS devices, wireless OBD-II scanners, wireless smart power meters) and a large number of applications (e.g. GreenGPS, BikeNet, Traffic Analyzer, Smart Attire) that utilize the various sensing devices motivate the need for a basic set of services organized in the form of an architecture which will ease the development of applications. The goals of our architecture are: (i) Collection, storage, analysis, and sharing of the sensor data, (ii) Plug-and-play support for a variety of sensing devices, (iii) Privacy preservation of individuals sharing sensor data, (iv) Grassroots impact, and (v) Easy application development.

Our architecture should enable easy collection, storage, analysis, and sharing of sensor data generated by everyday sensing devices. It should support plug-and-play of various sensing devices in an easy manner. Further, we are interested in an architecture that will support wide deployment and one that can be utilized by the common person, that is we want our architecture to have a *grassroots* impact. Since, sensor data can reveal sensitive information about an individual (for example, GPS sensor data may reveal an individual's home, work place location and the times they are away from home), privacy preservation should be integral to our architecture. Another goal that is of interest is ease of application development, our architecture should empower developers by providing generic tools that can be used for composing new human centric sensing applications.

The problem of designing a unifying architecture for human centric sensing applications has not been addressed earlier. An initial approach towards participatory sensing has been presented in *Partisans* [86]. Although, this architecture is still in its design phase and has not been implemented. Further, this work addresses only data verifiability and assumes the presence of a trusted third party for achieving privacy.

In contrast, we design and implement the first unifying architecture for supporting human centric sensing applications and provide a set of generic tools that address various challenges.

We adopt a client-server approach, where the clients are individuals collecting and sharing sensor data (possibly perturbed). The (aggregation) server on the other hand aggregates data from multiple clients and enables novel community sensing applications. This design is motivated by the wide success of the Web, where servers host data and clients *download* it from the servers. Our approach extends this popular technique by enabling the clients to *upload* data to the servers.

The modules are collectively built as an Internet application layer service that utilizes several Web application layer standards, such as XML and HTTP. The use of standards for the development of the PoolView modules enables the ease of their deployment and integration with existing Internet applications. The PoolView architecture with its various modules is illustrated in Figure 2.1.

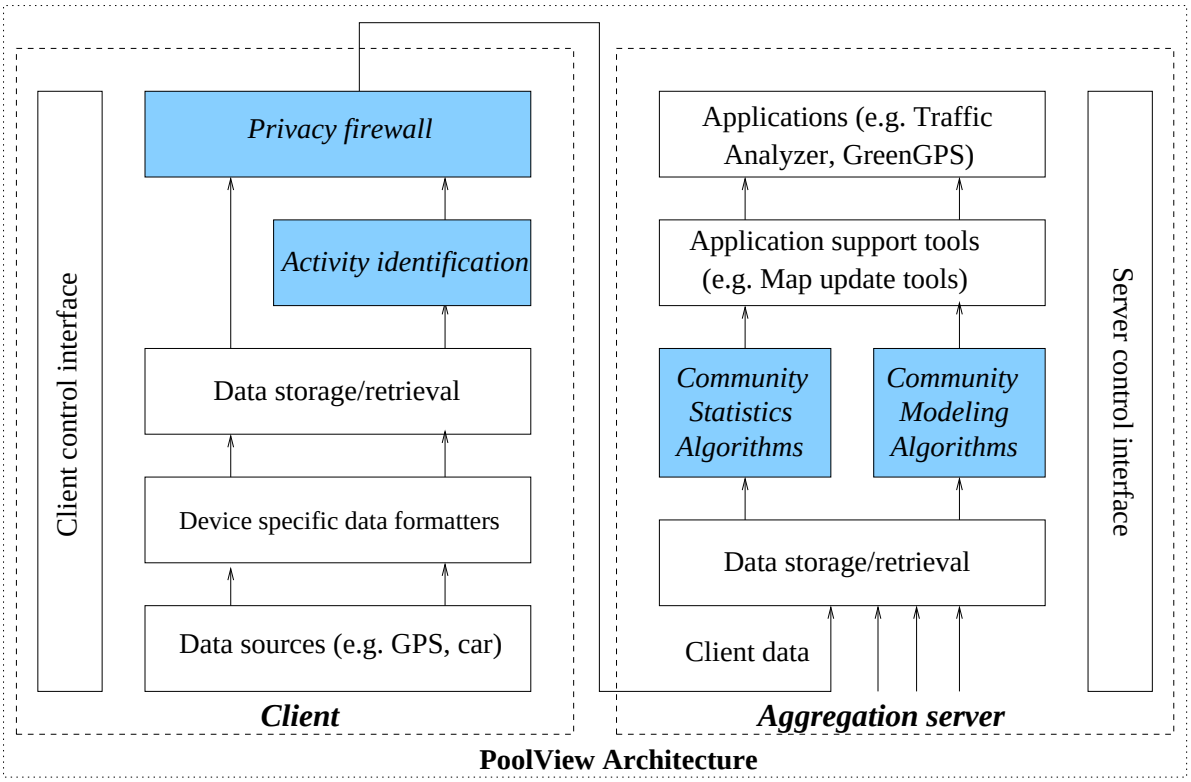


Figure 2.1: PoolView architecture (shaded modules are generic research challenges)

We note from Figure 2.1 that the architecture is divided into *client* and *server* parts. The various modules on the client side are: (i) Data formatters, (ii) Data storage/retrieval, (iii) Activity identification, and (iv)

Privacy firewall. The modules on the server side are: (i) Data storage/retrieval, (ii) Community statistics, (iii) Community data modeling, and (iv) Application support tools. The parameters for various modules can be controlled using the client and server control modules.

We will first give an overview of our architecture, both from the client side and the server side. On the client side, an individual with various sensing devices connects them to the PoolView's client side interface and uploads the collected data, which is then formatted as a standard XML stream and stored in the individual's private storage. These sensor data may then be analyzed to identify the physical activities performed by the individual and the identified activities may be stored back in the private storage. Finally, the sensor data collected can be shared, possibly in a perturbed fashion, with one or more aggregation servers.

On the server side, the sensor data shared by the individuals are stored on the aggregation server. These sensor data can then be analyzed to obtain various community related statistics. For example, the server can obtain traffic statistics in a given city from GPS location data shared by several individuals. Another example is where the server computes weight statistics of a population from shared measurement data. When the phenomenon that needs to be captured is complex and the sensor data available are sparse, our server provides prediction modeling techniques that can capture these complex phenomena. The server also provides map based application development tools, which can be utilized to display the statistics (or the raw sensor data) on maps.

2.1 Data Stream

The sensor data stream is a well-formed and valid XML [111] document. It describes a sensor data stream (being generated by various devices) in a device independent format, thus standardizing the representation of sensor data across multiple devices. XML is widely popular for sharing structured data across the Internet and is an extensible language that allows users to define their own tags, hence we choose XML for PoolView data stream's message body.

In what follows, we will describe the data tags that are associated with a data stream or data item. The primary goal of communication between various data sources and storage servers (client) or between the client and the aggregation server is the exchange of structured sensor data streams (or sensor data items).

The data fields that are superscripted with a $*$ are optional data fields, and may be left empty. The data fields that are superscripted with a ψ are fields that may be left optional on a conditional basis. The conditions under which these fields are optional are described at the end of this section.

`userid`

Unique string that identifies the user on the client side storage or aggregation server.

`object_type`

The type of object that is generating the data item. For example, shirt, smartphone, car.

`sensor_location`$^{\psi}$

The location of the sensor. For example, on left arm, inside car, in left trousers pocket.

`sensor_modality`$^{\psi}$

The type of sensor. For example, GPS, accelerometer, temperature.

`algorithm_type`$^{\psi}$

The type of algorithm that was used to generate this data item. The algorithm type is applicable only when the sensor data are processed. For example, a HMM based algorithm that takes input as accelerometer data streams from multiple sensors and outputs the activity of the person (such as walking, sitting, cooking, eating).

`start_time`

The time at which the data item began to be acquired. This data field indicates the start time for the data item generated if the given data item was obtained over a duration of time.

`end_time`*

The time at which the data item was generated. This data field indicates the end time for the data item generated if the given data item was obtained over a duration of time.

`data_value`

The value of the data item.

`data_unit`

The unit associated with the data value.

`sampling_frequency`*

The rate at which an individual data item is being generated for a data stream.

`latitude`*

Location latitude where the data item was generated.

`longitude`*

Location longitude where the data item was generated.

`privacy`

Field which specifies the type of privacy preserving technique applied (can be null if no privacy technique is applied).

`how`*

Attributes associated with the data item. For example, walking fast and driving erratically.

`confidence_interval`*

A confidence interval that provides the accuracy of the `data_value`. For example, a 10% error in the sensor data value generated by the accelerometer.

The fields that are marked as optional on a conditional basis distinguish raw sensor data from processed high level data. Any data stream that is in its raw form must define the fields, `sensor_location` and `sensor_modality`. Such a data stream may leave the fields `algorithm_type` and `data_tag` empty. Further, a processed data stream must define the fields, `algorithm_type` and `data_tag`. But, this type of data stream may leave the `sensor_location` and `sensor_modality` fields empty.

2.2 Client - Data Formatters

The client data formatters module is responsible to standardize the sensor data generated by varied devices. This module along with the standardized XML representation (Section 2.1) of the data stream abstract away

the device specific data format, thus enabling varied devices to connect to PoolView without the application developers being aware of the actual device itself. For example, developers simply query the data storage module for GPS sensor data and do not worry about which device (e.g. in-car GPS devices, smartphones, MicaZ motes) actually generated the sensor data. Each device has a specific sub-module that converts the device specific formatted data to an XML data stream, described in Section 2.1. In our implementation, an individual typically uploads sensor data collected using the PoolView's client control interface (which is a Web based interface) to their private storage server. During upload, the individual specifies the device used for data collection. The private storage server first processes the uploaded data by instantiating the correct sub-module (based on the device type) and generates an XML data stream. This module and the corresponding sub-modules are implemented in Java, each sub-module corresponds to a specific Java *class*. The various devices connect to this module using the Internet combined with a PHP and CGI-Perl based Apache module that instantiates the correct Java sub-module (depending on the type of the device chosen by the user).

2.3 Client - Data Storage

The client data storage module is responsible for storing and retrieving sensor data, which is represented as the standard XML data stream. This module enables individuals to keep a record of their sensor data, thus enabling them to share or analyze sensor data as and when new applications become available. For example, if a newer version of activity identification module become available that can identify activities better, it can be easily integrated into PoolView. We implement the data storage module as a *mySQL* server, which is a standard open source database service. One can also imagine the mySQL server being replaced by a secure and private cloud computing storage services, such as the ones provided by Amazon, IBM. The choice of a standardized database service enables us not to reinvent the wheel and queries (simple and complex) can be posed to our system using standard query languages such as SQL. These queries are encapsulated in an XML request, which are interpreted by a thin layer on the mySQL server. The query encapsulation approach decouples the actual SQL server implementation from PoolView's data storage/retrieval. For example, mySQL can be replaced with any other popular implementation of SQL. It also gives us the flexibility of developing complex requests for storing/retrieving sensor data streams, which could be inefficient using standard SQL

queries.

2.4 Client - Activity Identification

As alluded to in the Introduction (Chapter 1), activity identification is an important problem in several applications that are human centric, where the goal is to identify the physical activity performed by the individual (carrying the sensing device). We dedicate the discussion on activity identification and the challenges encountered to Chapter 3. Here, we will describe the APIs and functionality of this module. The client activity identification module is responsible for identifying the physical activities performed by individuals in their everyday lives. These activities could be *simple* such as walking, running, and sitting or *complex* such as cooking, eating, and hygiene. This module takes as input one or more raw sensor data streams (from the same type of sensor or multiple types of sensors) and identifies the corresponding physical activity performed by the individual. Activity identification, as discussed in Section 1.1, is necessary to enable various health care, entertainment, and personal record keeping applications. The current PoolView implementation identifies simple activities such as walking, running, and sitting when accelerometer sensor data are input (to the activity identification module). When microphone and accelerometer sensor data are input to this module, it identifies complex activities such as cooking, eating, and hygiene. The output (of this module) is a sensor data stream of time tagged physical activities, which is typically stored back in the database. We discuss the details of the activity identification algorithms (challenges faced and solutions proposed) in further detail in Chapter 3. This module is implemented in C with a Java based interface. The Java interface allows for easy integration with the rest of the modules and the C implementation enables a faster execution environment.

2.5 Client - Privacy Firewall

The client's privacy firewall module is the center piece of our architecture, as it controls the release of a user's private data to the outside world. The basic function of the privacy firewall is to screen or perturb user data in such a manner as to preserve the privacy of the data streams that the user owns. The privacy firewall is necessary as sensor data can reveal private information regarding individuals (when they share the data with

third parties). One can opt to not share their data, but we assume that the individual wishes to share their data to gain a certain value from it. As we have seen earlier in Chapter 1, privacy poses an interesting research challenge, which will be addressed in detail in Chapter 4. We will now describe the functionality and the implementation details of this module. A *privacy table* is the central data structure of the firewall. It can be thought of as a two dimensional array whose dimensions are (i) aggregation services and (ii) data types. A cell corresponding to a given service and data type contains a pointer to the corresponding perturbation model (currently, PoolView supports data perturbation for privacy). We discuss the details of the nature of perturbation to be applied in Chapter 4. The perturbation model is specified in a standard XML file. The privacy firewall module is implemented in Java, which takes as input the XML file (perturbation model) and XML formatted sensor data (generated by a query to the client data storage module) and generates as output perturbed sensor data as an XML data stream.

2.6 Server - Data Storage

Similar to the client side data storage module, this module supports the storage and retrieval of sensor data streams that are shared by individuals with the aggregation server. The main difference is that each individual sensor data stream is tagged with the user credentials (e.g. a unique *username* assigned by the aggregation server to individuals subscribing to it). Individuals share XML formatted sensor data (possibly perturbed to preserve privacy) with the aggregation server, which tags the data with user credentials and stores it in its data storage. The server side data storage is implemented as a mySQL server with an XML interface similar to the client side storage module. It is not hard to replace the mySQL storage services with a secure and private cloud computing storage services (as discussed in the client's data storage module). Again, we note here that the existence of a storage module provides the flexibility for new applications to evolve.

2.7 Server - Community Statistics

Participatory sensing applications rely on sensor data collection by individuals and sharing it among themselves to map common phenomena or compute community statistics. For example, individuals record GPS

sensor readings from their daily commutes and share it within a community to compute traffic related statistics (in the given city) [49, 61]. This module allows for the computation of various statistics from the shared sensor data, even if the data are perturbed. As we will show later in Chapter 4, this module implements a reconstruction algorithm that computes various community statistics when the sensor data is perturbed (according to the algorithm that will be described in Chapter 4). The input to this module is sensor data (possibly perturbed) in a standard XML format (which typically is provided by the data storage module on the server) and the required statistics specified as a well formatted XML document. The output is XML formatted data describing the results for the statistics (which were specified in the input). This module is implemented in Java and Matlab with a Java based API.

2.8 Server - Community Data Modeling

In the previous Section, the community statistics module can compute various statistics related to sensor data collected by individuals. Another important problem is when the data collected are sparse and we are trying to capture a complex phenomenon. For example, in order to obtain a complete traffic map of a large city, individuals need to contribute a large number of traffic measurements within the city (in order to obtain complete coverage). Typically, initial deployments will be sparse as the usefulness of the application is realized. As discussed in Chapter 1, the research challenge is to generalize well from the sparse high-dimensional sensor data to capture the complex phenomenon. We will discuss a solution approach to this problem in Chapter 5. The function of this module is to generalize from sparse high-dimensional sensor data to capture the complex phenomenon. The initial input to this module is multi-dimensional sensor data, various attributes that model the phenomena, and the model structure. It then builds prediction models using the approach we will describe in Chapter 5, which can be used to predict the phenomenon where the data are absent. The inputs are specified as well formatted XML files, the sensor data specified as per Section 2.1. The output is also a well formatted XML file with the predicted sensor data. This module is implemented in C++ and Matlab with a Java interface for easy integration with the rest of the modules.

2.9 Server - Application Support Tools

A large number of participatory sensing applications rely on maps to achieve their functionality. For example, CarTel [61] uses maps to achieve routing based on real-time congestion levels, Traffic Analyzer [49] uses maps to display traffic information on Google maps, BikeNet [35] presents bike routes on maps, CenceMe [76] maps individual's location information, and GreenGPS [46] utilizes maps for computing fuel efficient routes. Most of these applications rely on crude interfaces to existing map systems, such as Google Maps, MapQuest for their functionality and are restricted by the APIs provided by these services. This results in extremely inefficient solutions [61, 46]. For example, routing algorithms are proprietary and lack an API to modify the parameters used for routing, which translates into applications like CarTel and GreenGPS being extremely hard to implement using existing map based services. We address this concern by providing map based tools that empower the developer with APIs that allow for access to updating and modification of the maps. The solution combines various open source software with APIs that were developed by us. The map of a given area is maintained as an *OpenStreetMap (OSM)* [82]. OSM is the equivalent of Wikipedia for maps, where data are collected from various free sources (such as the US TIGER database [103], Landsat 7 [78], and user contributed GPS data) and an editable street map of the given area is created in an XML format. The OSM map is essentially a directed graph, which is composed of three basic object types, *nodes*, *ways*, and *relations*. A node has fixed coordinates and expresses points of interest (e.g. junction of roads, Marriott hotel). A way is an ordered list of nodes with tags to specify the meaning of the way, e.g. a road, a river, a park. A relation models the relationship between objects, where each member of the relation has a specific role. Relations are used in specifying routes (e.g. bus routes, cycle routes), enforcing traffic (e.g. one way routes). We provide a Java based API that can add new relations (which can be used to specify new statistics) or update them. For example, one can add a new relation that specifies average speed on different ways (streets), which can be used to compute fastest routes more accurately [61]. We will show in Chapter 5.3 that these tools can be used to compute fuel efficient routes.

We also provide a Java based interface to geocoding tools that translate street address inputs into latitude/longitude pairs. Geocoding is the process of finding corresponding latitude/longitude data given a street address, intersection, or zipcode. The actual geocoding process is implemented in Perl.

2.10 Conclusion

In this chapter, we developed an architecture and a set of modules that comprise PoolView. We showed that PoolView is an Internet application service that utilizes several standards such as XML and HTTP. We further described individual components, their functionalities and interfaces. The architecture provides support for the development of individual layers in an independent manner, thus allowing for easy extensibility. Each layer abstracts away its functionality and provides a clean standardized interface to interact with it. We envision that PoolView will provide a platform for future human centric sensing applications. Table 2.1 describes various applications (Smart jacket, smartphone, traffic analyzer, weight watchers, and GreenGPS) that were developed using PoolView and the corresponding components that were utilized.

Application	Smart jacket	Smartphone	Traffic analyzer	Weight watchers	GreenGPS
C - Data storage	✓	✓	✓	✓	✓
C - Activity ident.	✓	✓			
C - Privacy firewall	✓	✓	✓	✓	
S - Data storage			✓	✓	✓
S - Comm. stats.			✓	✓	
S - Comm. model					✓
S - App. supp. tools			✓		✓

Table 2.1: Applications and the corresponding PoolView components

We observe from Table 2.1 that several applications utilize multiple overlapping components, thus exemplifying the utility of PoolView and its generalizable nature. Note that, the smart jacket and smartphone applications utilize only the *client* portion of the modules, as they are personal sensing applications.

Chapter 3

Human Activity Identification

This chapter presents the details of the client side activity identification module of PoolView along with its use in in two different applications, a *smart jacket* embedded with MicaZ sensor nodes and a *smart phone*. We are motivated by the widespread availability of sensing devices in everyday lives of users, which will give rise to a new category of services termed as *personal monitoring* services. These services are software services that enable the monitoring of daily human activities, in the long term, short term, and real time. Such services monitor daily human activities through the collection and analysis of sensor data from devices that interact with their user on a daily basis. Examples of such devices include cell phones and clothes embedded with sensing devices. These services will find uses in several application domains, such as healthcare, social networking, entertainment, and personal record keeping. An example healthcare application is one in which a patient needs to be monitored at home for a prolonged period of time by a care-giver. Safety of people can be improved by providing services which automatically notify health-care providers in real-time during events of emergency (such as seizures, strokes, or accidents). Novel services that maintain records of personal activities are feasible. For example, jogging enthusiasts can keep track of their schedules and be able to answer short-term queries such as, "How much time did I spend jogging in the past month?". Such personal records can also help in providing medical care, such as detecting early onset of diseases. Entertainment services that answer questions such as, "Where was I on the Christmas eve of 2005?" or "Was I in Olive Garden when I last visited New York?" are feasible. Further, sensor information sharing can also be utilized to compute community-wide statistics. An example is where individuals share their speed information to compute aggregate traffic statistics.

A major research challenge to enable personal monitoring services is to be able to identify the physical activities (e.g. sitting, running, eating, cooking) performed by an individual. Human activity identification has been extensively addressed in the past [41, 91, 92, 96]. Most of these use specialized devices or lab based

environments to achieve human activity identification. The use of specialized devices allows for tuning of the hardware to achieve application specific optimization, whereas devices such as smartphones are not optimized for identification of activities. Such devices are equipped with possibly low quality sensors or are constrained by the device's primary functionality. Further, an important goal is that of transparency to the user, where the device monitors activities of the individual with minimal intervention to their daily lives. In this chapter, we develop a framework for identification of activities that utilize everyday existing devices (e.g. smartphones, clothing embedded with embedded devices).

We will begin by providing a high level overview of how activity identification is achieved and then provide details of its application to two different prototypes. Our activity identification framework combines *feature extraction* with *Hidden Markov Models*, a Bayesian learning technique to achieve activity identification. Our framework can identify basic activities such as *walking, running, and typing* when only accelerometer sensor data are input. The fusion of accelerometer and microphone sensor data results in the identification of complex activities such as *cooking, eating, and hygiene*. The basic idea in a Bayesian learning approach is to build a model for each activity that we wish to identify using *training* data for that activity. The input to build models can be various features such as energy, peaks, and entropy of the sensor data stream (in a particular time window). Once the models are built for each activity using the training data, future sensor data streams are matched against each of the models and the best match is identified as the activity corresponding to the input sensor data stream. The challenge in such an approach is to identify the right set of features that would significantly differentiate the activities. Further, it is also important to identify the right Bayesian learning approach. Several Bayesian learning approaches exist , such as classification tree, Naive Bayes, k-nearest neighbors [56] with varying levels of complexity. Our approach is to utilize a learning method that models time series, which naturally captures different activities (because human activities are time based and modeling them using static techniques will not work, as we will show later). This intuition justifies the choice of using Hidden Markov Models (HMMs), which is a natural choice for time series modeling. We will first briefly describe HMMs followed by the implementation details of each prototype and the evaluation of the activity identification framework.

3.1 Hidden Markov Models

A Hidden Markov Model (HMM) is a statistical model where the system being modeled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters from the observable parameters, based on this assumption.

A HMM, which is pictorially depicted in Figure 3.1 is characterized by the following parameters:

- N: The number of hidden states
- M: The number of distinct observation symbols per state
- $A_{N\times N}$: State transition probability distribution
- $B_{N\times M}$: Observation symbol probability distribution for each state
- $\Pi_{N\times 1}$: Initial state distribution

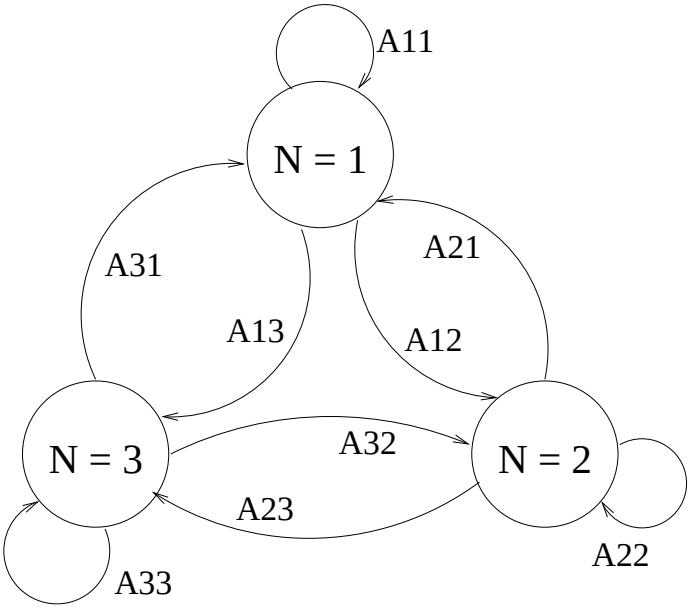


Figure 3.1: Pictorial representation of a 3-state HMM

A HMM has two phases, a *learning* phase and a *testing* phase. We will describe these two phases in further detail in the following two sections.

3.1.1 HMM: Learning Phase

In the learning phase of HMM, given the observation symbols (in this case, the training data generated by the sensor data for each activity), the problem is to efficiently compute model parameters. More formally, the model parameters, (A, B, Π), need to be adjusted to maximize the probability of the observation sequence. There are several approaches to this estimation [89], but a popular approach is the *Baum-Welch* technique. The Baum-Welch technique is an iterative method and is derived from the Expectation-Maximization (EM) algorithm. The basic idea is to reestimate the model parameters in an iterative manner. At each stage, a new model is derived and compared with the existing model based on how well it generates the observation sequence. Details regarding the exact mathematical derivations can be found in [89].

3.1.2 HMM: Testing Phase

In the testing phase of HMM, given a model and observation sequence, the problem is to compute the probability with which the given model generates the observation sequence. This can be thought of as a scoring scheme, where we are trying to infer which of several models generate a given observation sequence. This relates to activity identification as follows, the sensor data stream generated (the activity corresponding is unknown) is processed to extract features, which form the observation sequence. This feature observation sequence is then matched with several models (one for each activity) to identify the model (and the activity) that generates the given sequence with maximum probability. The observation sequence probability, given a model is computed using the *Forward-Backward* procedure, which is explained in further detail in [89].

HMMs have been used in several machine learning and speech recognition applications [89]. In the context of activity identification [77, 108], HMMs have been used to identify complex wood workshop activities. In contrast to this, our emphasis is on identifying a broader range of common every-day activities.

3.2 Smart Jacket

In this section, we will describe the development of smart jacket, a heavy winter jacket embedded with MicaZ motes, which record human activities and location information using 2-axis accelerometers and GPS, respectively. These measurements are stored locally (on the flash memory of MicaZ motes) until they can

be uploaded (to a base mote, a MicaZ mote attached to a PC). A typical application scenario for the usage of the smart jacket is as follows. An individual wearing the smart jacket goes about their normal daily activities as usual over the course of a day. During that time, the jacket records sensory data pertaining to the owner's whereabouts and activities. When the system comes in the vicinity of the base mote, the logged data is uploaded reliably to a private repository associated with the person. This record can potentially act as a memory aid or help doctors in augmenting a patient's clinical information. Figure 3.2 gives a typical usage scenario of our system.

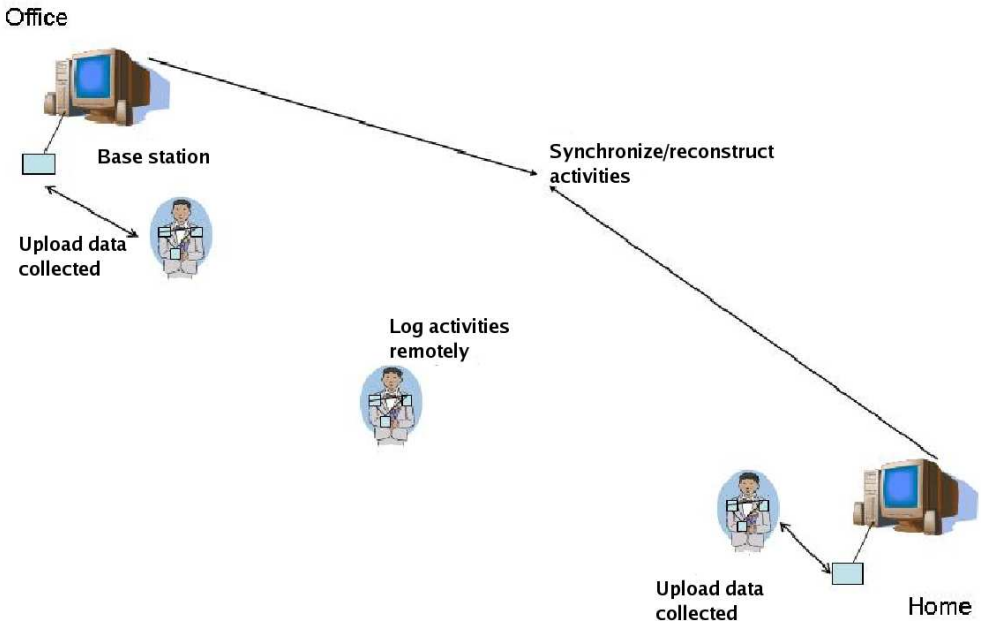


Figure 3.2: A typical operational scenario

We will briefly discuss the problems faced while implementing the smart jacket prototype and the corresponding solutions. The main research challenge that we face is to identify the human activities from the acceleration sensor data, which we will discuss in Section 3.2.2

3.2.1 Problem Discussion and Implemented Solutions

The implementation issues can be classified as follows, (i) data collection and storage, (ii) data upload, (iii) data synchronization, and (iv) power management. The major research challenge is the identification of basic human activities.

Data Collection and Storage

In a typical operational scenario, the system will collect data periodically and store it in the flash memory. A MicaZ mote has 512 KB of flash memory, which is used for data recording purposes. Hence, we observe that a single sensor sampling at the rate of 30 Hz, generating 2 bytes per sample will consume the flash memory in approximately four hours. Increasing the number of sensors used will consume the flash even faster! A simple proposition to reduce the amount of flash consumed is to reduce the sampling rate, but this would be inadequate as the data values recorded cannot be used to identify the activities. We conducted simple experiments to identify an ideal sampling rate, and found that a sampling rate of $25\ Hz$ suffices for activity identification. This sampling rate requirements calls for other methods to reduce the amount of flash consumed in order to increase the disconnected time of operation of the system.

We propose two different methods to reduce the amount of flash used without loss of the precision of data collected. Both are different data compression algorithms based on the observations we made during the deployment of the system. The first method, termed the *truncate filter* is based on the observation that, for normal human activities, the least significant eight bits of the ten bit output is sufficient. This doubles the disconnected time of operation of the system. The second method takes advantage of the fact that we do not need to record any data values when the clothing is still, as there is no activity taking place. Similar to run-length encoding, at the end of a stillness interval, a special separator value is inserted in the log, indicating that the jacket has been still and the number of samples for which it has been still is recorded. This method is termed the *stillness filter*.

Data Upload

An important part of the system is to upload the data collected to a server through an base mote. Three separate issues are to be addressed as part of the upload protocol, which are as follows:

- The rate at which the upload occurs, which directly affects the amount of flash available.
- The upload transparency, which relieves the user from the hassle of pressing a button or consciously making a gesture to upload data.

- Reliability of upload, Loss of data packets can lead to incorrect interpretation of activities or lack of data for certain periods of time. As expected, when the user moves away from the base mote, the packet reception probability goes down.

We develop a new protocol optimized for our application scenario, which meets the above goals.

Our protocol combines ideas from various data dissemination protocols like *Deluge* [60] and *PSFQ* [106]. It achieves the goal of reliable, transparent and fast upload. Transparency is achieved by using a *beaconing* scheme. The base sends beacons periodically, which are ACKed by the motes in the system, if the motes are in the range of the base. Reliability is achieved using a NACK scheme. We tweaked the payload size in TinyOS and the number of packets sent every second, to come up with an optimal data rate to send data as fast as possible. We use the CSMA MAC protocol which comes with the TinyOS networking stack. Our protocol makes sure that only one mote is communicating with the base at a given time to minimize collisions and increase throughput. This is ensured as follows. When the base gets replies from the motes, it elects a single mote (mote X) on a First-Come-First-Serve basis and sends a *send_data* packet to this mote. After sending this packet, the base enters a state where it ignores further beacon replies. If it does not receive any data packets from mote X within a specified timeout period, it resets its state to send beacons.

The protocol also ensures a fair channel allocation mechanism, so that starvation does not occur. This is ensured as follows. When the motes receive a beacon from the base, they start timers that are inversely proportional to the time they have not won an election. Thus, the mote that has not won an election for the longest time will (most likely) send a reply first to the base.

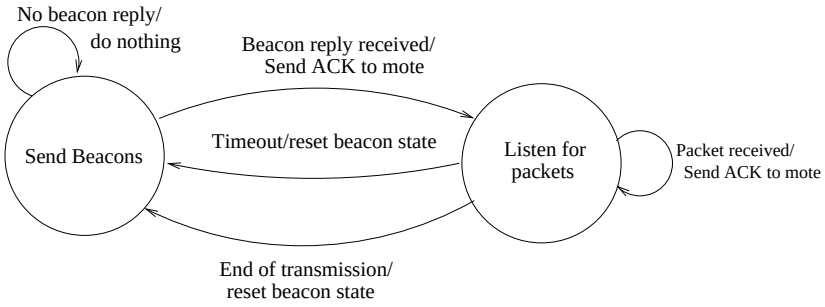


Figure 3.3: State diagram for the base

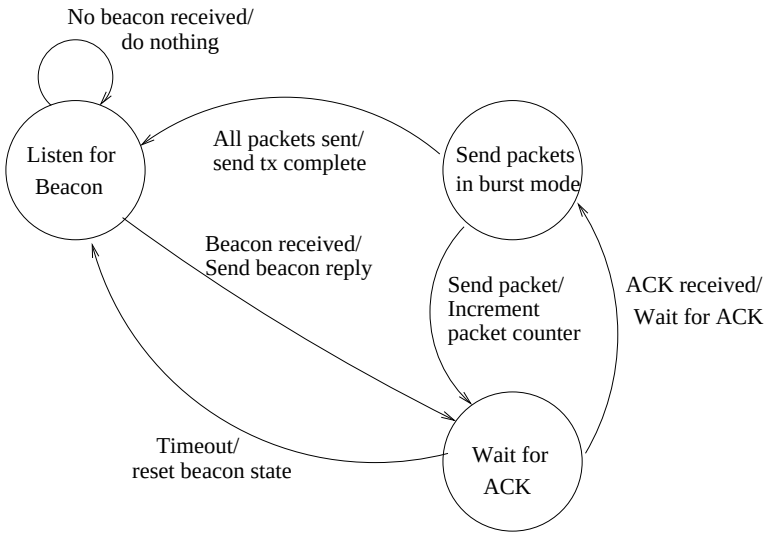


Figure 3.4: State diagram for the motes on the person

Figures 3.3 and 3.4 summarize the upload protocol which we present in the form of state diagrams. The annotations on each arrow have the form *X/Y*, where X indicates the event which has occurred and Y describes the action to be performed on occurrence of this event.

Data Synchronization

We need a mechanism by which it is possible to correlate the activities recorded on different parts of the body. This problem can be termed as the *data synchronization* problem, where each data item collected needs to be temporally correlated with data items collected on other motes.

This problem has been addressed in [112]. The scheme presented in [112] needs a base station in the vicinity of the data collection nodes, which SATIRE cannot use, and thus a new data synchronization scheme is needed. Apart from the above scheme, there have been several time synchronization protocols which synchronize the clocks on motes [73], [45]. However, recording absolute time values leads to a considerable overhead in the flash. The periodic message exchanges contribute an additional overhead.

To maintain temporal correlation among the data values collected on different motes, a *leader* mote sends out beacons which are used to synchronize data streams on the different motes in the network. Each beacon is identified by a beacon number. When a beacon is received, the associated beacon number is recorded in real-time in the flash mid-stream along with a separator, to differentiate them from data sample

values. On the PC, identically numbered beacons are aligned to the same time reference. The only overhead in our method is that of recording the values of beacons in the flash. Beacons samples occur several orders of magnitude less frequently than data, which makes their overhead acceptable.

Power Management

The typical lifetime of a mote which is *on* for the entire period of time is about seven days. With continuous logging and radio communication, the lifetime may be further reduced. Replacement of batteries every week is cumbersome and cost ineffective. Hence, a power management scheme is necessary to extend the lifetime of the system. An acceptable design goal for a seasonal outer garment in our opinion is to last for about three months (i.e., the entire season).

We propose to use a simple duty cycle based scheme. In this scheme, a mote goes to *sleep* after it detects a brief period of stillness. It wakes up after *n* seconds and checks whether or not stillness continues. If the mote determines that it is in motion, it starts logging data. Otherwise, it goes to sleep again. During this cycle, the mote keeps track of the amount of time it has been sleeping and logs this information when the stillness interval terminates.

In our current jacket prototype, we observed that the jacket was still for 90% of the time. If we assume a 5% duty cycle during low-power operation, the lifetime of the system can be extended seven times, as can be seen from Equation 3.1. In Equation 3.1, P_d is the average power consumption when the mote operates as described above, P_n is the average power consumption when the mote is active all the time, and d is the duty cycle (in our case, it is 0.05). The power consumption of the mote in a low-power state is assumed to be negligible. This gives us $\frac{P_d}{P_n}$ to be about seven, which translates into an increase in the lifetime of the jacket from one to about seven weeks. This is close to the season-long goal we set out for our smart jacket.

$$P_d \quad = \quad 0.1 \times P_n + 0.9 \times (d \times P_n) \tag{3.1}$$

As for the GPS mote, it draws a higher current than a normal sensor and will last for about half a day if it is left on continuously. The GPS mote takes about seven seconds to obtain a fix (usually). In the active state, the GPS mote obtains a fix every minute and sleeps for the rest of the minute. When the jacket is still, it does not obtain a fix. In this scenario, the GPS mote lasts for seven weeks ($\frac{60}{0.1 \times 7} \times 0.5$ days), which is

close to our season-long goal.

3.2.2 Basic Human Activity Identification

Our goal in this section is to identify basic human activities, which include *sitting, writing, typing, walking*, and *cycling*. We will first describe and evaluate a popular approach to human activity identification and then show the drawbacks of this approach. We will then evaluate our HMM based activity identification framework.

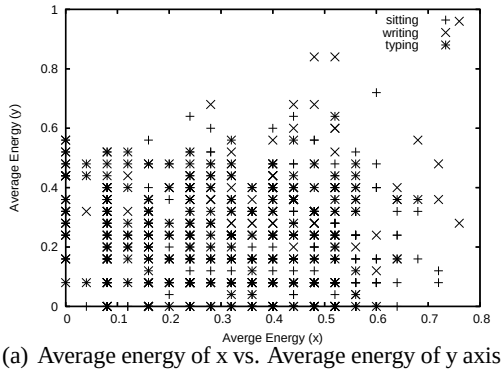
Feature Vector based Identification

A popular approach for the identification of basic human activities is a feature vector based approach, where a set of features are extracted from the accelerometric signal and used to identify activities which are well spread out in the feature space. An example of such an approach is described in [96], and we claim no novelty in this regard. Feature vectors have also been used in speech recognition [98].

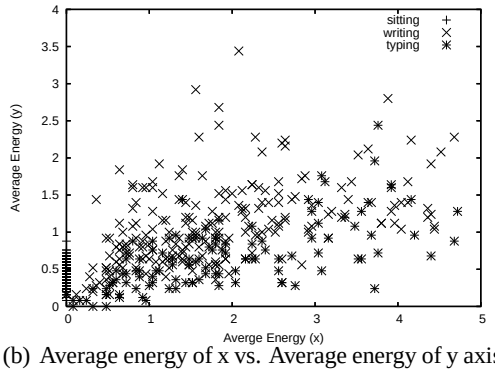
Several features of a signal have been introduced in [96] and [105] for the purpose of activity identification. These features can be mapped onto a multidimensional feature space which can be used for activity identification. An example of a feature is the energy of the difference signal of the x and y accelerometer axes. Figure 3.2.2(a)-(f) plots a two-dimensional feature space for different activities, where each dimension is the energy of the difference signal of the corresponding accelerometer axis.

Figures 3.2.2(a)-(e) plot the x and y axis energy values for five motes placed on the lining of the jacket, each for three activities, namely `s`itting, `w`riting, and `t`yping. We observe that these activities have overlapping regions in all the five motes. From our evaluation reported in [47], we observe that these activities are not clearly discernible even when using several features. However, an activity such as `w`alking has a clearly identifiable region in this two dimensional feature space, as shown in Figure 3.2.2(f).

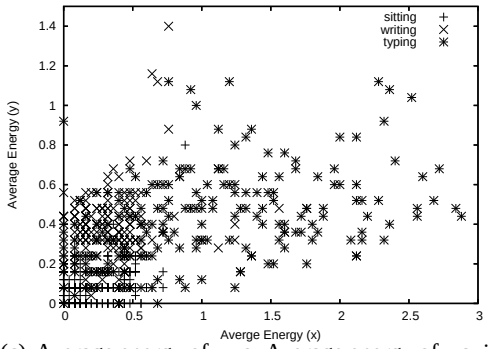
The above graph plots the activities over two features only. Since, it is not viable to pictorially represent an activity in more than 3 dimension space, we opt to show the activities in a two dimensional space. But, in our implementation, the feature space consists of several other features such as average, standard deviation, root mean square, range, integral, *temporal variation*, and *rotational direction*. We showed in [47] that, even though we use a large number of features, the accuracy of identification of the activities by the feature vector



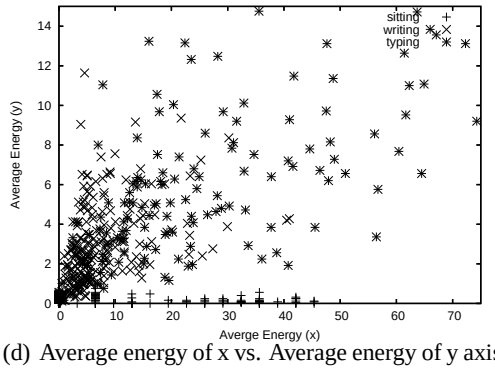
(a) Average energy of x vs. Average energy of y axis for mote 1 for sitting, typing and writing



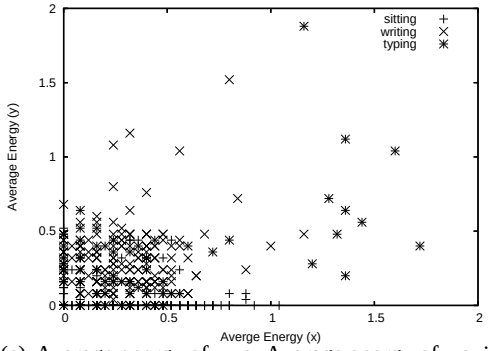
(b) Average energy of x vs. Average energy of y axis for mote 2 for sitting, typing and writing



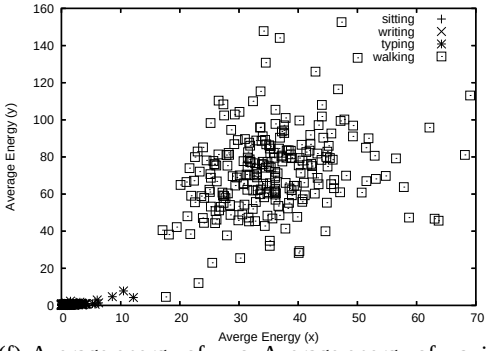
(c) Average energy of x vs. Average energy of y axis for mote 3 for sitting, typing and writing



(d) Average energy of x vs. Average energy of y axis for mote 4 for sitting, typing and writing



(e) Average energy of x vs. Average energy of y axis for mote 5 for sitting, typing and writing



(f) Average energy of x vs. Average energy of y axis for mote 3 including walking

Figure 3.5: Figure showing average energies for various activities

based method is poor.

Temporal variation is the sum of absolute Euclidean distances between accelerometric vectors of any two successive time instances. Rotational direction gives an idea of clockwise/anticlockwise rotational movement during the activity [96].

An activity is represented as a static n-dimensional vector, where n is the number of features, obtained through representative training data. To identify a given activity, features are extracted from the raw accelerometric data, and a least error match with the representative feature vectors is found.

HMM based Identification

We found that the accuracy of the feature vector approach was poor when we used it to identify multiple activities, as shown in the previous section. To overcome the drawback of using a static feature vector model, we use a dynamic Hidden Markov Model (HMM) to solve the problem.

In our current implementation, the observation sequence that is used as input to the HMM is obtained by calculating two features, namely the energy and range of the difference signal. We found this approach of using features as input to the HMM to be more accurate than using just the raw accelerometric data. The difference signal is defined as the signal obtained by taking the difference between consecutive values of the signal. Energy of the difference signal for each axis is the sum of squares of the values of the difference signal for that axis, as shown in Equation 3.4. Range of the difference signal for each axis is the difference between the maximum and minimum values of the difference signal for that axis, shown in Equation 3.4.

$$Energy_x = \sum_{t=t_i+1}^{t_i+\tau} (x_t - x_{t-1})^2 \tag{3.2}$$

$$Range_x = max_{t=t_i}^{t_i+\tau}(x_t) - min_{t=t_i}^{t_i+\tau}(x_t) \tag{3.3}$$

$$where\ x\ is\ any\ axis,\ the\ window\ is\ (t_i, t_i + \tau)$$

Each value of the features, energy and range, is computed over a window of values of the difference signal. We found that using the features, energy and range, we were able to identify activities with reasonable accuracy. An exploration into other features that can be used to further improve the accuracy is needed.

These features are first extracted from the raw accelerometric data. For each feature, we use an empirically observed range of values, and uniformly map values within this range on to a set of observation symbols unique to that feature and belonging to a fixed alphabet. The size of this fixed alphabet is M. For each feature, we consider one second windows of observation symbols for each mote and axis. We then concatenate the corresponding one second windows (by time) of observation symbols across all motes and axes,

and then across all features to obtain the observation sequences. We thus obtain an observation sequence for each second of activity. These observation sequences are used as input to the hidden Markov models.

As discussed earlier, there are two phases to using a HMM. The first is a training phase, where the HMM for an activity learns the model parameters that maximize the probability of observing a representative data set for that activity. A training set of observation sequences for each human activity is used to learn the HMM parameters that characterizes the given set with the highest probability. The second is the inference phase, where given the hidden Markov models for the activities and an observation sequence to be classified, the model which matches the given observation sequence with the highest probability is inferred.

To solve the problem of human activity identification, we use an ergodic (every state of the model can be reached from every other state) and discrete observation HMM. For details of the training and inference techniques, the reader is referred to [89]. For each activity, we have a $N = 10$ state HMM. We use a total of 355 observation symbols ($M = 355$) per state. We use the well known Baum-Welch technique, which is based on an Expectation Maximization (EM) algorithm, for training the HMM for each activity. To identify an activity, we use a Forward-Backward procedure, which given a HMM model, estimates the probability that the observation sequence is generated by that model. Using this procedure, the probability that the observation sequence is generated by the HMM for each of the activities is calculated. The activity that yields the highest probability is chosen as the activity represented by the observation sequence.

Evaluation Results

We classify the activities into *low-energy* and *high-energy*. An activity is classified as *low-energy*, when the energy of the difference signal (as described in Section 3.2.1) summed over a period of time is lower than a threshold. Otherwise, the activity is classified as *high-energy*.

We compare the accuracy of our HMM based approach with that of the feature vectors. In all our activities, the data set was obtained by conducting each activity three times, each for a period of five minutes. A sample of one minute was used to train the HMM for each activity. Ground truth was verified by manually recording the activity at a given time instant. Data sets were obtained for two different users.

Figures 3.6 and 3.7 plot the accuracy of detecting a set of three *low-energy* and two *high-energy* activities using both the feature vector and the HMM approaches, for user 1 and user 2, respectively. The

activities considered were `stillness`, `typing`, `writing`, `walking`, and `cycling`. The feature vector approach performed poorly when compared to the HMM approach. This is due to the fact that the feature vector approach does not consider the sequence in which the motion is performed, but rather relies on a set of static features. The results in our experiments for the HMM based approach were obtained by defining a *confidence metric*. For a given input data, identification of the activity involves generating the probabilities with which the different HMM models (one for each activity) match this input, and choosing the model with the highest probability match. Let p_i be the probability that the input matches HMM i, $1 \leq i \leq n$, where n is the number of HMM models (activities). Let p_j be the highest probability. For a confidence metric of θ, the given input is classified as belonging to model j, if the following is true:

$$\frac{p_j}{\sum_{i=1}^{i=n} p_i} \geq \theta$$

If the above does not hold, the given input is classified as not recognizable (shown as the activity `''I dont know''` in Figure 3.8). In all our experimental results, we use a confidence metric of $\theta = 0.8$.

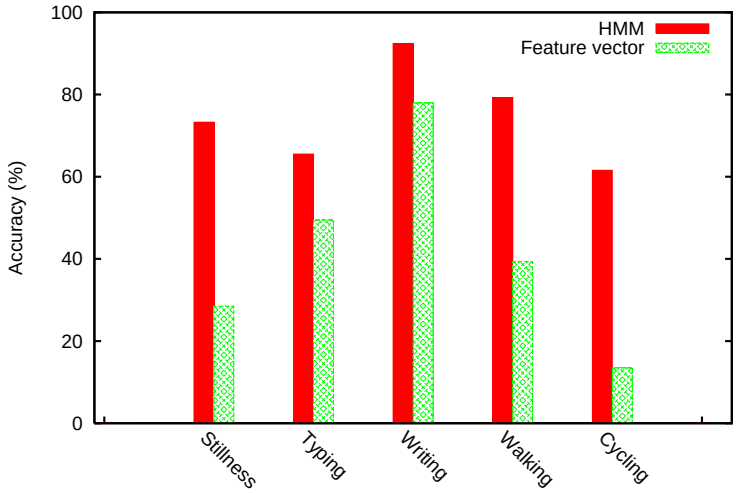


Figure 3.6: Accuracy of activity identification using HMMs and feature vectors for user 1

Figure 3.8 plots the activity over time for an experiment lasting 350 seconds by user 1. We observe from this figure that the user was `still` (sitting) for 150 seconds, after which he walked for about 75 seconds. He then started writing.

We then trained the HMM using one user's data and tested the accuracy of identifying the other user's

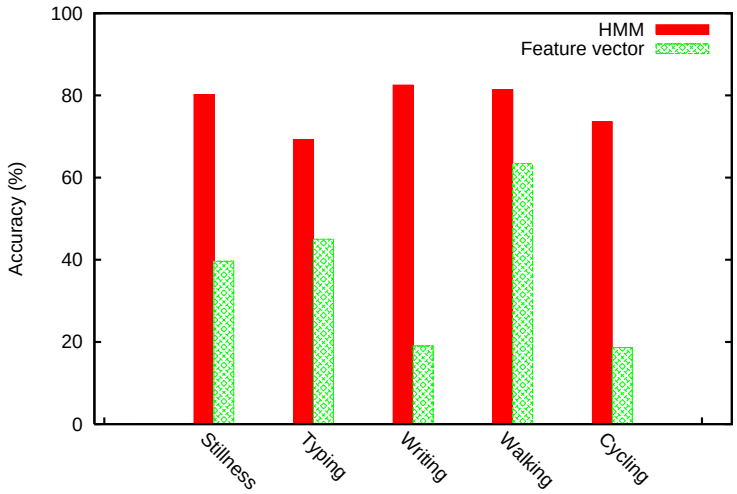


Figure 3.7: Accuracy of activity identification using HMMs and feature vectors for user 2

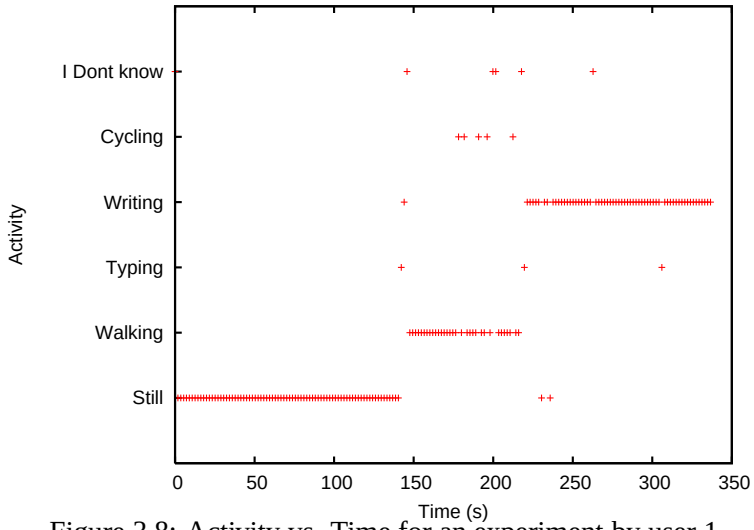


Figure 3.8: Activity vs. Time for an experiment by user 1

activities. This is plotted in Figure 3.9. We observe that certain activities have very high accuracy, for example, user 1's data on user 2's training set gives high accuracy for `stillness`, `writing`, and `walking`. Whereas the activities `typing` and `cycling` fared poorly.

In such a case, a new user can use the jacket to identify a set of pre-trained activities. To identify new activities or to improve the accuracy of identification of existing activities, the user can specially train the jacket to suit their needs.

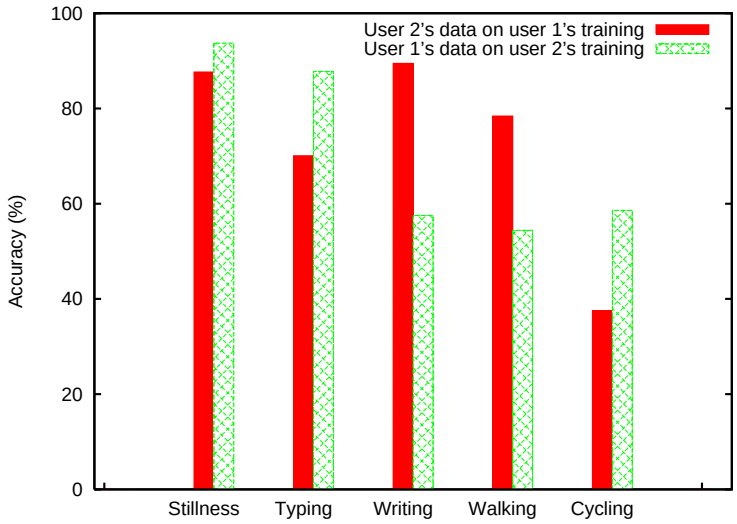


Figure 3.9: Accuracy of activity identification without user specific training

Location Tracking using GPS

The GPS mote can be used to track the location of the user when he or she is not occluded from the GPS satellites. We conducted experiments to track the location of a user. For one such experiment, the location details and the speed-time plots are shown in Figures 3.10 and 3.11, respectively. From these figures, we were able to deduce that the user was walking in `Region 1` (with an approximate speed of 1.5 m/s) and was still for about 5 minutes in `Region 2`. At about a time of 60 seconds, we observe that the speed of the user was 0 m/s. This was because the user was waiting to cross a busy road. The user's speed was found to fluctuate between 0 and 10 m/s during times 700 and 1150 seconds, indicating that the user was in a vehicle that made frequent stops. In fact, this was found to be true as the user was traveling in a campus bus, which made frequent stops. From time 1150 seconds onwards, the user was found to be walking at about 1.5 m/s.

3.3 Smartphone

A most common personal device that people own is a cellphone. With the advent of sensing devices being integrated in smartphones, they are transforming into a personal sensing device. For example, today's smartphones are often equipped with micro-electromechanical (MEMS) sensors, in particular accelerometers, which have a small form factor and low power consumption (this is in addition to the traditional sensing modalities of the cellphone, the microphone and camera). And most of the smartphones have GPS built-in.

Figure 3.10: Expt. 2: Location tracking using GPS

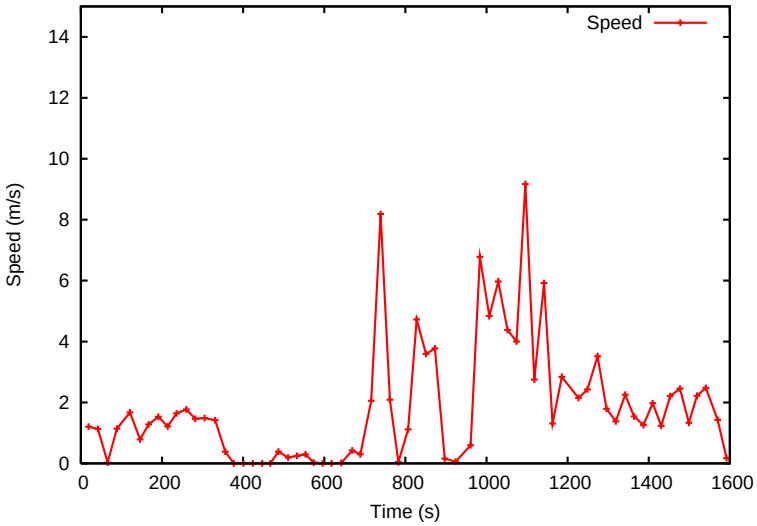


Figure 3.11: Expt. 2: Speed vs. Time of user

The growing popularity of smartphones and its multimodal functionalities makes the device an individual's personal proxy, a context aware device, an activity inference device, and even a payment proxy [107].

We develop a personal monitoring service using the smartphone (here, the smartphone is acting as an activity inference device). In particular, we will extend our previous approach for activity identification in two ways. One is to use multimodal sensing and the other is to expand the set of activities identified from a basic set (e.g. walking, typing) to more complex activities (e.g. cooking, brushing teeth). In particular, we are interested in the identification of *activities of daily living* (ADL). Examples include cooking, desk work,

brushing teeth, and so on.

Smartphones are quite prevalent these days and their capabilities have also increased multifold in the past few years. Examples of such smartphones equipped with various sensors include the Nokia N-series (N82, N95, N96), Apple iPhone, and the BlackBerry. Many of these smartphones are equipped with location, motion, light, audio, and video sensors. Since Nokia provides a large number of smartphones that have a common operating system and APIs, we choose to use the Nokia N95 for our work.

We design and implement a general software architecture for the purpose of data collection on the Nokia N95 (which can be utilized to collect data from other Nokia phones that have the required hardware). The N series of Nokia phones use a client-server based operating system, the Symbian OS [81], designed for resource constrained mobile devices. Access to lower level hardware is provided through a request callback sequence, where servers (abstractions of lower level hardware) respond to requests from clients.

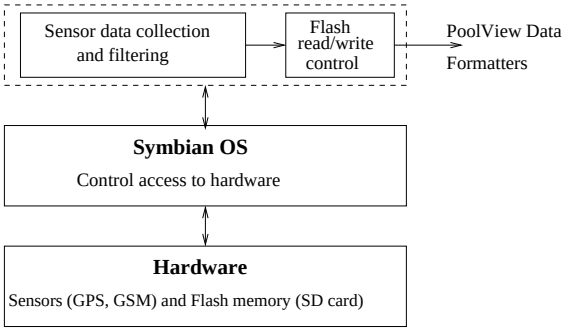


Figure 3.12: Figure depicting the various components of our software design on the cellphone using Symbian OS.

Figure 3.12 shows the components of our software architecture, which enables a generic and flexible collection of data from various sensors. We can see from Figure 3.12 that the components fall into three categories, the lowest level includes the hardware of the cellphone (microphone, GPS, accelerometer), abstractions of which are provided by the server components of Symbian OS. The modules in the application layer (top most) provide three main functionalities: (i) Data collection from the server components, (ii) Recording data collected from the sensors, and (iii) Reading data recorded from the flash for upload to a PC for data analysis. Our architecture is modular, flexible, and extensible and enables data collection from various sensors with ease. A snapshot of the data collection application that utilizes the above architecture is shown in Figure 3.13. The application allows for tagging the data streams being recorded with the

corresponding activity (chosen by the user from the drop down list).

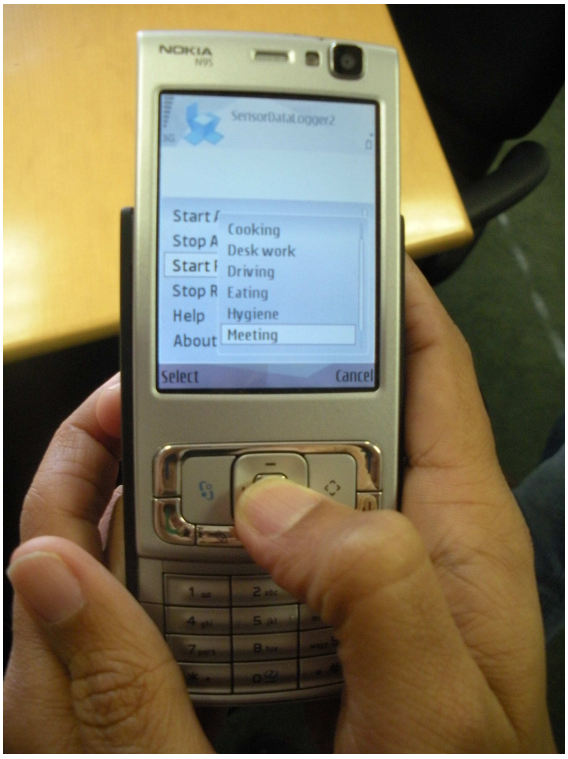


Figure 3.13: Figure showing the application for activity tagging

Our current implementation records four different sensors: the microphone, accelerometer, GPS, and GSM (GSM information is used to determine the user's location when GPS signals are unavailable) for offline analysis. While our current inference algorithm is fairly light-weight, the initial development focuses on a proof-of-concept implementation of multisensor fusion for activity detection. In the future, we envision a real-time activity inference technique on the phone. Also, recording the identified activity may be useful for long-term trend analysis, as is shown by *smart attire* [47].

We developed a prototype that implements a data collection software on a popular smartphone (similar to the smart attire prototype), which records the microphone and accelerometer sensor data in the local flash memory of the smartphone. These data are later uploaded to the PC of the user, which is then analyzed using the HMM framework to identify the activity performed by the user. We will now describe the details of identification of ADLs.

3.3.1 ADL Inference Algorithm

In this Section, we describe the identification of activities of daily living (ADLs). We combine multi-sensory data to identify complex actions of users in their daily lives. In particular, we show that integrating the data from the microphone and the acceleration sensor embedded in a typical smartphone (Nokia N95) is a promising approach for ADL monitoring. We utilize the feature-space-combination approach described at the beginning of this Chapter1, in which we extract information from both sensors sampled at different rates. This is accomplished by a synchronous feature extraction approach in which features from each sensor are computed independently at the same, constant time-frame rate. The extracted features are fed to a computationally light-weight algorithm, suitable for implementation on a smartphone, such as the N95.

In our study, the user wears the phone on the waist. When a user performs an activity such as cooking, the data capturing routine within the smartphone is activated. The data collection module samples the acceleration sensor at 7 Hz and the microphone at 8 kHz using the inbuilt sensor APIs. This forms the input to a trained, ADL monitor within the activity identification module of PoolView.

We conducted an empirical study to obtain the training and testing data set for the automated classifier. Eight distinct ADLs and instrumental activities of daily living (IADL), as shown in Table 3.1, are considered in the present study.

Activity		
Name	Type	Activity Description Reference
Aerobic	Dynamic	Walking, running, lifting weights, etc.
Cooking	Dynamic	Food preparation, heating, grilling, etc.
Desk Work	Static	Typing, reading at desk.
Driving	Static	Driving in a car.
Eating	Static	Eating while seated or standing.
Hygiene	Dynamic	Washing dishes, brushing teeth, etc.
Meeting	Static	Present in, or attend a meeting.
Watching TV	Static	Watching TV while not performing any of the above activities.

Table 3.1: Activities of daily living considered within the empirical study

These activities comprise of both static and dynamic activities. An activity is “static”when there is no significant acceleration signal detected at the waist while performing that activity. A “dynamic” activity, on the other hand, contributes to significant acceleration while accomplishing the activity. These eight activities are chosen as they form the basic ADLs and IADLS [79, 88]. Note that the activities in Table 3.1 can not be

differentiated based on acceleration signals alone.

We use an hidden Markov Model (HMM), to model each activity. Since HMMs are well known models of time series and have been used successfully to recognize basic human activities (Section 3.2.2), we use it in the present study to identify ADLs/IADLs based on the fusing the information from sensed accelerometer and microphone data. The input to the HMMs consists of the following features derived from the sensed tri-axial acceleration data in overlapping time frames of 5 seconds with a 1.67 seconds frame shift. The first feature detects relative change in body orientation in xy-z plane with respect to a calibration phase when the user is presumed to be standing. The next feature is the 3-dimensional vector magnitude of acceleration. The magnitude of the 3-dimensional acceleration is related to the energy expended in performing a particular physical activity such as walking. We also compute the skewness of the magnitude of the 3-dimensional acceleration. Finally, we compute the entropy of the acceleration in the z-axis. Relative inclination helps in distinguishing activities that depend on whether a person is sitting (e.g., "Eating"), standing (e.g., "Cooking") and lying-down (e.g., "Watching TV"). Energy expenditure, skewness, and entropy help distinguish between dynamic activities (e.g., "Aerobic) and static ones (e.g., "Desk Work").

The microphone data is also processed at the same frame length of 5 seconds. We extract spectral shape features that are related to the audio content [70]. Specifically, the audio signal is first subjected to a frequency analysis in each time frame. The resulting speech spectra are processed by a Mel-frequency filterbank [70], comprising 26 triangle-shaped filters. The low frequency cut-off of the first filter is set to 0 Hz and the high frequency cut-off of the last filter was set to 4000 Hz. A log compression is then applied to the resulting spectra. Finally, the spectral coefficients are converted to cepstral coefficients via the discrete cosine transform [83]. These cepstral coefficients help in differentiating between different classes of dynamic activities (e.g., "Cooking" and "Hygiene"), or different classes of static activities (e.g., "Meeting" and "Driving"). We use the first 12 cepstral coefficients as they are known to be the most useful in describing the content of an audio signal [70].

3.3.2 Evaluation Results

We now present the descriptive results of our empirical study (see Section 3.3.1) in this section. We recruited eight male participants between the age groups of 20-37 years to participate as subjects for the empirical

data collection experiments. We instructed the users to go about their regular routine and perform their daily activities as naturally as possible and placed no restrictions on the location or time of the day. Users were encouraged to wear the device as much as possible for a period of eight weeks in either their pocket or a carrying pouch. We compensated each participant with a $\$20$ gift card to a local movie theater. All participants signed an user agreement that stated that they agree to the collection and use of microphone, acceleration, GPS (if available), and GSM (cell information that can be used to track location) data for research purposes.

In order to collect data to train the classification algorithm and validate the results of classification, we modified the data collection part of our software design to enable users to label their activities. Specifically, we instructed the users to label the beginning and the end of each activity. Additionally, when the phone is switched on, the user calibrated the acceleration axes by standing still for a period of 10 seconds. The start and end of calibration was cued by making the device vibrate.

A total of 80 hours of tagged activity data was collected. A partial data set of 45 hours is used for developing and training the automated classification algorithm. Eight activity-level HMMs are trained, one for each activity listed in Table 3.1. All have 3 states, whose output distribution is modeled as a mixture of 8 Gaussians. The 3 state model is chosen to model the 'transition-into', the 'steady state', and the 'transition-out' of each activity. Testing is performed on a 7 hour subset of the remaining data (i.e., data not including in training). The rest 28 hours of data is unusable because the users did not either calibrate the device or label the activities correctly. An HMM toolkit, HTK [116], is used for training. During testing, we perform a maximum-likelihood decoding to determine the most likely activity. This form of decoding could be viewed as a single finite state model composed of individual HMMs with transitions between various activities classes modeled as equally likely. We make this simplifying assumption currently due to lack of a large dataset to model transition between activities accurately. Note that recognition is user independent; the data from all the users are used to construct the HMMs and the testing does not exploit the knowledge of the user identity.

The results, in terms of accuracy of classification, are summarized in Table 3.2. Accuracy refers to the portion of time windows in which the classified and labeled activities match. The results indicate that the recognition accuracy is quite high for the following activities: "Aerobic,""Cooking," "Driving," and

"Hygiene'.' We find that "Eating" is hard to distinguish from "Watching TV" because a majority of "Eating" activity was performed when the user was "Watching TV" and hence the audio and acceleration features are quite similar. One way to alleviate this would be allow for "N-best" outputs from the classifier, where "N" refers to those outputs during decoding that exceed a threshold on their likelihoods. The low accuracy of "Meeting" on the other hand is due to the availability of a limited amount of our data corresponding to this class. Table 3.3 shows the results in terms of precision and recall. We observe from Table 3.3 that our precision results are also quite good, except for the "Meeting" class due to the limitation mentioned above. Overall, the results in Table 3.2 and Table 3.3 show the potential of combining the information from the acceleration sensor and the microphone for the identification of ADLs. However, further evaluation is required to confirm the statistical validity and significance of these preliminary results.

Activity	**Accuracy (%) 3-state**	**Accuracy (%) 1-state**	**Accuracy (%) 5-state**
Aerobic	82	79.3	83.1
Cooking	100	100	100
Desk Work	53	34	50
Driving	87.6	96	77
Eating	12.7	12.7	14
Hygiene	99	64	43
Meeting	12.7	12.7	14
Watching TV	88	87	88

Table 3.2: Performance of the automated ADL classifier

Activity	**Precision %**	**Recall %**
Aerobic	76.8	81.9
Cooking	76.4	100
Desk Work	50.8	53
Driving	100	87.6
Eating	51.2	12.7
Hygiene	65.6	99
Meeting	12.5	12.7
Watching TV	55.9	88

Table 3.3: Precision and recall of the classifier

We now present the confusion matrix in Table 3.4. As we mentioned earlier, our dataset consists of people "Eating" while "Watching TV", and hence both these activities were confused with each other. We

also observe that “Desk Work” and “Meeting” are confused with each other due to the open space work environment in which these data were collected.

Activity	Aerobic	Cook	Desk	Drive	Eat	Hygiene	Meet	TV
Aerobic	1197	145	0	0	120	0	0	0
Cooking	0	523	0	0	0	0	0	0
Desk Work	329	0	958	0	0	0	519	0
Driving	33	0	0	891	7	0	0	86
Eating	0	0	333	0	241	199	0	1117
Hygiene	0	0	0	0	4	380	0	0
Meeting	0	0	596	0	0	0	87	0
Watching TV	0	17	0	0	99	0	91	1523

Table 3.4: Table showing the confusion matrix

Finally, we provide an empirical justification regarding our choice of the 3-state HMM. Table 3.2 also shows the accuracy results when using a Gaussian Mixutre Model (a 1-state HMM) or a 5-state HMM for comparison. We can observe from Table 3.2 that in terms of the average accuracy, the 3-state HMM outperforms the 1-state and 5-state HMMs, thus supporting our choice of using the “transition-into”, “steady”, and “transition-out” states. We conclude that our choice of discriminative acceleration and audio features and a 3-state HMM provide a promising approach to identifying ADLs.

3.4 Integration with Facebook

The activity identification module is integrated with Facebook, a popular social networking application. Individuals can provide access to their activity log to friends on Facebook. This utilizes a simple privacy module (of PoolView’s privacy firewall), one that checks if a person accessing the individual’s activity log is their Facebook friend. This access is implemented using the Facebook API combined with PHP code on PoolView’s client side.

3.5 Conclusions

In this chapter, we presented a novel activity identification framework that identifies basic activities when only accelerometer data are input and complex activities when accelerometer and microphone data are in-

put. This framework forms the activity identification module of PoolView. We show the accuracy of this approach using two different prototypes, one is a smart jacket embedded with MicaZ motes and the second is a smartphone embedded with various sensors. We utilize a feature-space-combination approach in conjunction with a time-series Bayesian learning technique, the Hidden Markov Models (HMMs). We also showed through extensive evaluation results that our technique outperforms existing algorithms for activity identification. We also developed a simple privacy module that integrates the activity log with Facebook, a popular social networking application.

Chapter 4

Privacy Preservation

In this chapter, we focus on enabling community sensing applications. Community sensing (also called as participatory sensing, citizen sensing) is where individuals collect sensor data and share it among themselves to map common phenomena or compute community statistics. Earlier community sensing applications [22, 35, 61, 94] have focused on data collection and analysis aspects. An important aspect that was not considered in these applications was privacy. In this chapter, we will summarize the mathematical foundations (which was developed collaboratively) and develop service implementations to enable *grassroots* participatory sensing applications. We consider communities of individuals with sensors collecting streams of private data for personal reasons. These data could also be of value if shared with the community for fusion purposes to compute aggregate metrics of mutual interest. The main problem in such applications is privacy, which motivates the work in this chapter.

We are interested in addressing privacy assurances in the absence of a trust hierarchy. We rely on data perturbation at the data source to empower clients to ensure privacy of their data themselves using tools that perturb such data prior to sharing for aggregation purposes. Privacy approaches, including data perturbation, are generally met with criticism for several good reasons. First, it has been repeatedly shown that adding random noise to data does not protect privacy [66, 59]. It is generally easy to reconstruct data from noisy measurements, unless noise is so large that utility cannot be attained from sharing the noisy data. Second, anonymity (another approach to privacy) does not help either. Anonymized GPS data still reveals the identity of the user. Withholding location data in a radius around home can be a solution, but opting to withhold, in itself may reveal information. Moreover, in a sparsely deployed network, the radius would have to be very large to truly anonymize the data. A third question is whether the assumption of lack of a centralized trusted entity is justified. After all, we already entrust our cell phone providers with a significant amount of information. It should not be difficult to provide added-value services that benefit from the current (fairly

extensive) trust model.

Below, we address the aforementioned questions prior to presenting our approach. We address the problem of privacy in time-series data. The fundamental insight as to why perturbation techniques do not protect privacy is correlation among different pieces of data or between data and context (e.g., identity of owner). In what follows, we will present a technique that addresses the aforementioned problem of correlations within a data stream. We show that with proper tools, non-expert users can generate appropriately-correlated application-specific noise in the absence of trust, such that data of these individuals cannot be reconstructed correctly, but community aggregates can still be computed with accuracy. We further explain how non-expert users might be able to generate the appropriate application-specific noise without trusting external parties related to that specific application. Observe that inability to reconstruct actual user data largely obviates the need for anonymity. Our solutions are not needed for scenarios where a hierarchy of trust exists. In contrast to such scenarios, we are interested in providing a way for individuals in the community to collect information from their peers such as "how well does this or that diet or exercise routine work" or "what patterns of energy use at home really worked for you to reduce your energy bill"? Obviating the requirement to find a mutually trusted entity before data are collected is a way to encourage the proliferation of grassroots participatory sensing applications.

PoolView's client-server architecture is adopted, where clients share (perturbed) private sensory data and servers (called *pools*) aggregate such data into useful information made available to the community. PoolView presents a simple API for individuals to set up new pools the way they might set up a Wiki or discussion group. Simple APIs are also provided for clients to subscribe to pools and export their data. Interactions between clients and servers rely on the common XML based data-stream abstraction (Section 2.1). The stream allows an individual to share a sequence of (perturbed) data measurements such as weight values or GPS coordinates (a logged trip). The goals of the perturbation are: (i) to preserve the privacy of application-specific data streams against common reconstruction algorithms, (ii) to allow computation of community aggregates within proven accuracy bounds, and (iii) that perturbation (which may be application-specific) can be applied by non-expert users without having to trust the application. Hence, any person can propose a custom statistic and set up a pool to collect (perturbed) data from non-expert peers who can verify independently that they are applying the "right" (application-specific) perturbation to

preserve their privacy before sharing their data.

As alluded to above, ensuring privacy of data streams via perturbation techniques is complicated by the existence of correlation among subsequent data values in time-series data. Such correlations can, in general, be leveraged to attack the privacy of the stream. For example, sharing a single data value representing one's weight perturbed by adding a random number between -2000 and 2000 pounds will usually not reveal much about the real weight. On the other hand, sharing the current weight value every day, perturbed by a different random number, makes it possible to guess the weight progressively more accurately simply by averaging the sequence to cancel out noise. Perturbing the sequence by adding the *same* random number every day does not work either because it will reveal the trend in weight measurements over time (e.g., how much weight the individual loses or gains every day). The goal of the privacy preserving algorithm is to hide both the actual value and trend of a given individual's data series, while allowing such statistics to be computed over a community. Hence, for example, a community of weight watchers can record their weights as measured on a particular diet, allowing weight-loss statistics (such as average weight loss and standard deviation of loss) to be computed as a function of time on the diet.

To instantiate PoolView architecture's privacy preserving aspect, we have implemented and deployed two PoolView services (pools), one for computing average weight (Weight Watchers) of a self-selected community (e.g., all those on a particular diet), and another for computing traffic statistics (Traffic Analyzer) in a privacy-preserving fashion. We present data and results from these two applications. The rest of this chapter is organized as follows, we will first summarize the perturbation technique that achieves privacy (as described above). We will then utilize this privacy preserving technique in the applications, Weight Watchers and Traffic Analyzer, that highlight PoolView's privacy preserving and community statistics modules.

4.1 Time Series Data Privacy

In this section, we will briefly describe the perturbation method and the reconstruction algorithm that was developed in [49]. The perturbation problem is defined as follows. Perturb a user's sequence of data values such that (i) the individual data items and their trend (i.e., their changes with time) cannot be estimated without large error, whereas (ii) the distribution of community data at any point in time, as well as the average community data trend can be estimated with high accuracy.

For instance, in the weight-watchers example, it may be desired to find the average weight loss trend as well as the distribution of weight loss as a function of time on the diet. This is to be accomplished without being able to reconstruct any individual's weight and weight trend. For another example, it may be desired to compute the average traffic speed on a given city street, as well as the speed variance (i.e., the degree to which traffic is "stop-and-go"), using speed data contributed by individuals without being able to reconstruct any individual's speed and acceleration curves.

Examples of data perturbation techniques can be found in [5, 4, 39]. The general idea is to add random noise with a known distribution to the user's data, after which a reconstruction algorithm is used to estimate the distribution of the original data. Early approaches relied on adding independent random noise, which were shown to be inadequate. For example, a special technique based on random matrix theory has been proposed in [66] to recover the user data with high accuracy. Later approaches considered hiding individual data values collected from different private parties, taking into account that data from different individuals may be correlated [59] [1]. However, they do not make assumptions on the model describing the *evolution* of data values from a given party over time, which can be used to jeopardize privacy of data streams. This section describes a perturbation technique that specifically considers the data evolution model, where it is shown that the technique is strong against attacks that extract regularities in correlated data such as spectral filtering [66] and Principal Component Analysis (PCA) [59].

This technique perturbs data in such a manner that the privacy of time-series data (measuring some phenomenon, P) can be preserved if the noise used to perturb the data is itself generated from a process that approximately models the phenomenon. For instance, in the weight watchers example, we may have an intuitive feel for the time scales and ranges of weight evolution when humans gain or lose weight. Hence, a noise model can be constructed that exports realistic-looking parameters for both the direction and time-constant of weight changes. We can think of this noise as the (possibly scaled) output of a *virtual user*. The noise model generation and the trust implications are discussed in [49].

Once the noise model is available, its structure and probability distributions of all parameters are shared with the community. By choosing random values for these parameters from the specified distribution, it is possible, for example, to generate arbitrary weight curves of virtual people showing weight gain or loss.

[1] Since the data correlation is across individuals of the population, we will not compare it with our work in the evaluation section

A real user can then add their true weight curve to that of one or several locally generated virtual users obtained from the noise model. The actual model parameters used to generate the noise are kept private. The resulting perturbed stream is shared with the pool where it can be aggregated with that of others in the community. Since the distributions of noise model parameters are statistically known, it is possible to estimate the sum, average and distribution of added noise (of the entire community) as a function of time. Subtracting that known average noise time series from the sum of perturbed community curves will thus yield the true community trend. The distribution of community data at a given time can similarly be determined since the distribution of noise (i.e., data from virtual users) is known. The estimate improves with community size.

A useful refinement of the above technique is to separate in the virtual user model parameters that are *inputs* from those that express intrinsic properties of the model. For example, food intake may be an input parameter of a virtual user model. Inputs can be time-varying. The perturbation algorithm allows changing the values of input model parameters with time. Since the input fed to the virtual users is not shared, it becomes very hard to extract real user data from added noise (i.e., virtual user) curves.

One last question relates to the issue of trust. Earlier, we motivated perturbation approaches in part by the lack of a central trusted party that would otherwise be able to privately collect real unperturbed data and compute the needed statistics. Given that non-experts cannot be asked to program noise models for each new application (or even be expected to know what these models are), and since they cannot trust the data collection party, where does the noise (i.e., the virtual user) model come from and how does a non-expert client know that the model is not fake? Obtaining the noise model from an untrusted party is risky. If the party is malicious, it could send a "bad" model that is, say, a constant, or a very fast-changing function (that can be easily separated from real data using a low-pass filter), or perhaps a function with a very small range that perturbs real data by only a negligible amount. Such noise models will not perturb data in a way that protects privacy.

The answer comes from the requirement, stated earlier, that the noise added be an approximation of the real phenomenon. Incidentally, observe that the above requirement does not mean that the noise curve be similar to the user data curve. It only means that both come from a model of the same structure but different parameters. Hence, in the weight example, it could be that the user is losing weight whereas the

noise added is a curve that shows weight gain. Both curves come from the same model structure (e.g., a first order dynamic system that responds to food intake with a gain and time-constant). The models would have different parameters (a different gain, a different time-constant, and importantly a different input modeling the time-varying food intake).

With the above in mind, we allow the server (that is untrusted with our private data) to announce the used noise model structure and parameter distribution to the community of users. The model announced by the server can be trusted by a user only if that user's own data could have been generated from some parameter instantiation of that model with a non-trivial probability. This can be tested locally by a curve-fitting tool on the user's side the first time the user uses the pool. Such a general tool is a part of the client-side PoolView's privacy preserving module. Informally, a noise model structure and parameter distributions are accepted by a user only if (i) the curve fitting error for user's own data is not too large and (ii) the identified model parameter values for user's data (that result from curve fitting) are not too improbable (given the probability distributions of model parameters). The formal mathematical details of the above technique can be found in [49]. We will now discuss the application of this technique to Traffic Analyzer (Section 4.2) and Weight Watchers (Section 4.3).

4.2 Traffic Analyzer

The traffic analyzer case study is motivated by the growing deployment of GPS devices that provide location and speed information of the vehicles that they are deployed in. Such data can be used to analyze traffic patterns in a given community (e.g., average speed on a given street between 8am and 9am in the morning). Analysis of patterns such as rush hour traffic, off-peak traffic, average delays between different key points in the city as a function of time of day and day of the week, and average speeding statistics on selected streets can shed light on traffic safety and traffic congestion status both at a given point in time and historically over a large time interval.

With the above in mind, we present a case study that evaluates a traffic analysis application in the context of privacy preservation, where we study the performance of the perturbation techniques (applied in the context of traffic analysis). We picked two main streets whose traffic characteristics we would like to study for illustration. To emulate a community of users, we drove on these streets multiple times (in our

experiments, five graduate students took turns driving these streets at different times of day). We collected data for a community of 30 users. We used a Garmin Legend [50] GPS device to collect location data. The device returns a track of GPS coordinates. The sampling frequency used in our experiments was 1 sample every 15 seconds. Each trip represented a different user for our experimentation purposes. The stretch of each of the two roads driven was about 1.3 miles. Data was collected in the morning between 10 am and 12 noon as well as in the evening between 4 pm and 6 pm.

In a more densely deployed system, the assumption is that data will be naturally available from different users driving over the period of weeks on these city streets at different times of day. Such data may then be shared retroactively for different application purposes. For example, individuals interested in collecting data on traffic enforcement might collect and share speeding statistics on different city streets or freeways they travel (e.g., what percentage of time, where, and by how much does traffic speed exceed posted signage). Such statistics may come in handy when an individual travels to a new destination. Since speeding is a private matter, perturbation techniques will be applied prior to sharing.

In our deployment, each user shares their data using PoolView's client side interface. An individual collects location and speed information using in-car GPS devices (e.g. Garmin nuvi, TomTom have GPS trace recording capabilities). The recorded sensor data is then transferred to a PC and uploaded to PoolView's secure and private storage service. This data is then perturbed according to the algorithm described in the previous Section. The user also has an option of viewing the perturbed data in a graphical format and comparing it with the original data. The user can then share the perturbed data with an aggregation server (which provides the Traffic Analyzer service). The aggregation server collects GPS sensor data from the 30 users. It divides city streets into small segments of equal length. The average speed on each segment is then calculated from perturbed user data. We will first describe the noise model and how it is generated.

Generating the Noise Model

In order to employ the perturbation scheme described earlier, we need a noise model. Since the GPS data is collected with a very low frequency (1 sample every 15 seconds), speed may change dramatically on consecutive data points. Figures 4.1 shows the real speed curve of one user on Green street in the morning. We model the speed curve of each user as the sum of several sinusoidal signals (observe that any waveform

can be expressed a sum of sinusoids by Fourier transform). For simplicity, we choose to use six sinusoids that represent the common harmonics present in natural speed variations of city traffic. The noise model is therefore as follows:

$$f(k) = a_0 + \sum_{i=1}^{6} a_i \sin(b_i * k + c_i) \tag{4.1}$$

The speed model in Equation (4.1) is characterized by 19 parameters. Once the model for the speed is obtained, we need to model the distribution of all 19 parameters such that the speed stream generated by this model has the same dynamics as the real speed curves. The service developer will collect a few speed measurements empirically (which is what we did), take that small number of real speed curves, and use an MMSE curve fitting to find the range of each parameter. This approach is used by us to obtain the distribution of the parameters. The distribution of each parameter was then chosen to be a uniform within the range obtained. A sample of speed curve is shown in Figure 4.1.

Having produced an approximate noise model, the aggregation server announces the model information (structure and parameter distribution) to the users (PoolView clients). Participating users use this information to choose their private noise parameters and generate their noise streams using client-side software (which includes a generic function generator in the privacy firewall). Each user's individual speed data is perturbed by the given noise and sent to the aggregation server when the user connects to the server. Typical perturbed data is shown in Figure 4.1.

Reconstruction Accuracy

In order to compute the community average, noise distributions at each time instance, k, must be available for the aggregation server. Obtaining the exact noise distribution at each time k given the parameter a_i, b_i, and c_i can be difficult. Therefore we approximate the noise distributions by generating a numbers of noise curves (10000 samples) following the model in Equation 4.1 and compute normalized histograms of noise values at each time instance. These histograms are approximations of noise distributions.

To show reconstruction accuracy using the community reconstruction method (refer to [49] for the mathematical details of reconstructing community data), the computed community average speed curve for each street is presented in Figure 4.2. Even with a very small community population (17 users), the community

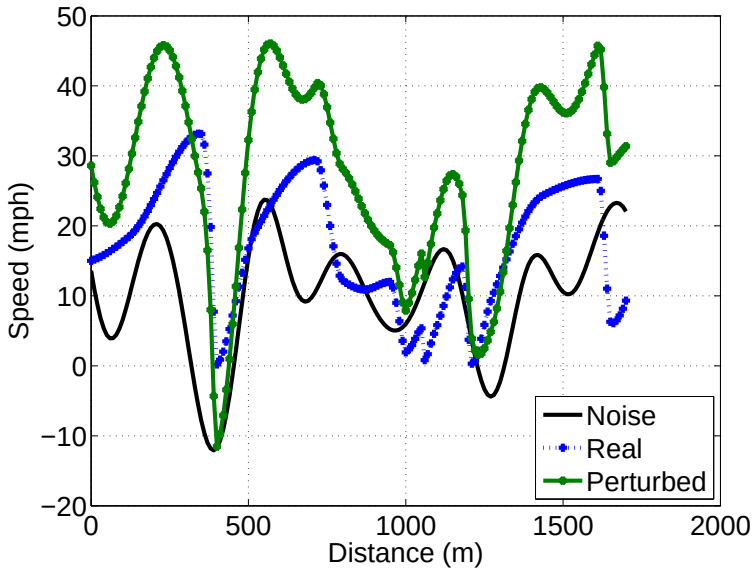


Figure 4.1: Graph showing the real speed, noise, and perturbed speed curves for a single user

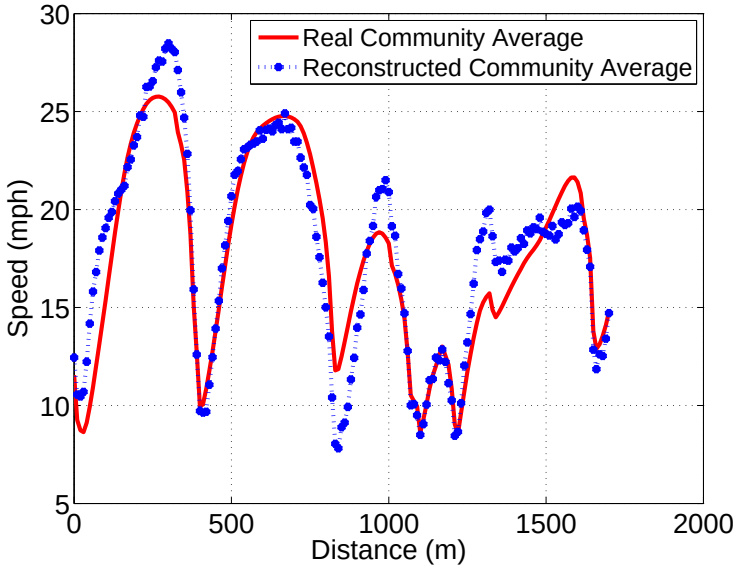


Figure 4.2: Graph showing the reconstructed community average speed vs. distance for a population of 17 users

average reconstruction still provides a fairly accurate estimate (the average error at each point is $1.94\ mph$).

We also illustrate the PoolView's Traffic Analyzer aggregation server interface that displays the average speeds on two streets in Figure 4.3. This instantiation of PoolView's aggregation server uses the map based support tools (described in Section 2.9).

Next, we examine the reconstruction of the community speed distribution at a given location on Uni-

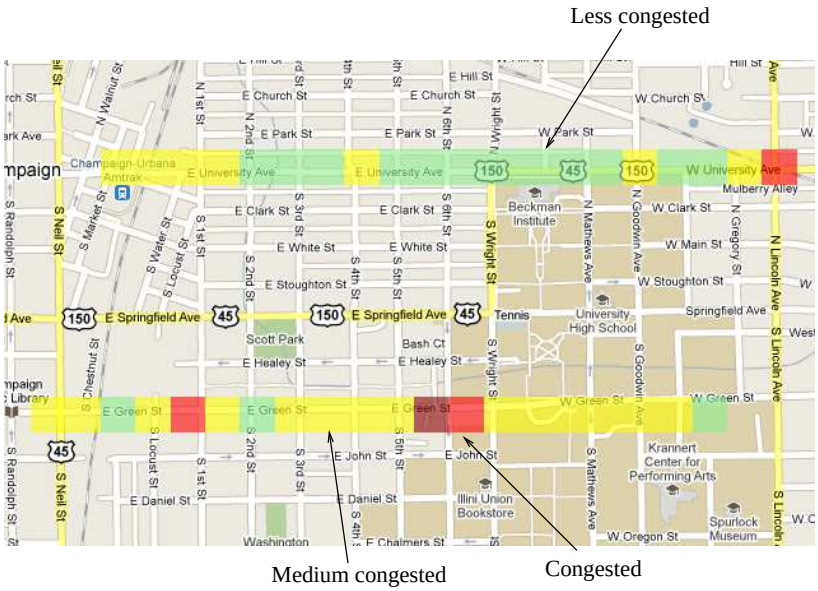


Figure 4.3: Figure showing the traffic analyzer aggregation server side interface

versity Avenue. Since our actual collected data was limited, we emulated additional user data by doing random linear combinations of data from real users. The real community speed distribution is shown in Figure 4.4(a). The reconstruction method presented in [49] is used to estimate the community speed distribution from the perturbed community data, with the result being shown in Figure 4.4(b). We note from both these figures that the distributions are quite similar, showing that even with a small community of users (17 users), we can accurately reconstruct the entire distribution.

4.3 Weight Watchers

The Weight Watchers case study is motivated by the numerous weight watchers and diet communities that exist today. An individual on a particular diet monitors her weight on a periodic basis, perhaps by taking a weight measurement once a day. This individual would likely be interested in comparing her weight loss to that of other people on a diet in order to get a feedback regarding the effectiveness of the diet program she is following. Although, the person would like to do it in such a manner that her weight data remains private.

In the Traffic Analyzer application, to the extent of the authors' knowledge, there is no good speed model for a vehicle on a city road. Thus, the speed is modeled in a semi-empirical way. However, in many other applications, accurate data models are well known and hence can be used to provide more privacy. The

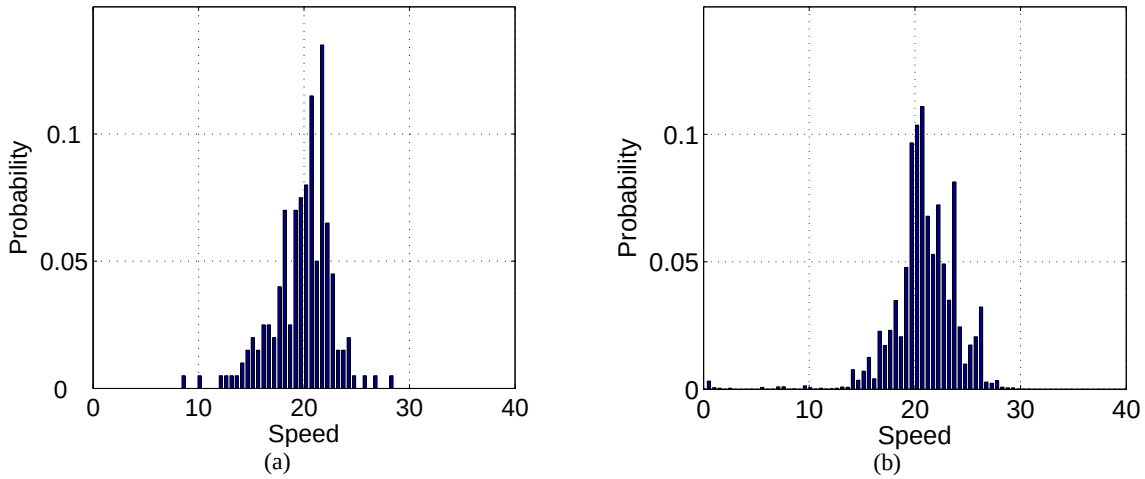


Figure 4.4: Figures showing the real (a) and reconstructed (b) community speed distributions for a population of 200 users

Weight Watchers application is one such example. Several models for weight loss and dieting have been proposed in existing literature [6, 23, 42, 75]. We adopt the model proposed in [75], which is a non-linear model, where the weight of a dieting user over time can be roughly approximated by three parameters: λ_k, β, and W_0. β is the body metabolism coefficient, W_0 is the initial weight of the person right before dieting, and λ_k is the average calorie intake of that person on day k. The weight $W(k)$ of a dieting user on day k of the diet is characterized by a non-linear equation:

$$W(k) = W(k-1) + \lambda_k + \beta W(k-1)^{3/4} \tag{4.2}$$

$$W(0) = W_0 \tag{4.3}$$

The equations 4.2, 4.3 are used to generate the noise stream.

In our deployment, we recorded the weight of a single user over the course of sixty days, once each day. We generate the parameters for a typical user based on the data from our deployment and use these to emulate multiple users.

The parameters for this model include λ_k, β and W_0. The range of λ and β can be found in [75]. The range of the initial weight W_0 can be taken as the weight of a normal adult which is from 80 pounds to 210 pounds. The simplest distribution for these parameters is uniform within their respective ranges. Samples

of the real weight data, noise and the perturbed data are shown in Figure 4.5.

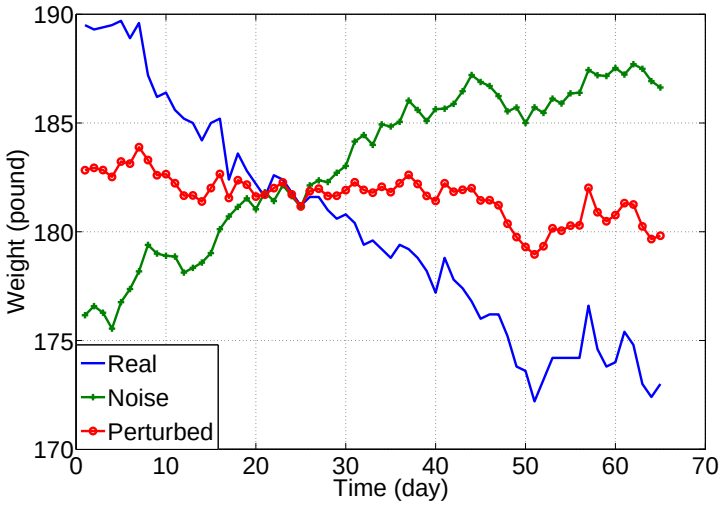


Figure 4.5: Graph showing real weight, noise, and perturbed weight of a single user

In this application, we demonstrate a different way of perturbing the user data, but use the same algorithm to reconstruct the community distribution. Given the generated noise n, and the data x, the perturbed data is generated as follows, $y = Ax + Bn + C$. In this type of perturbation, A, B and C are random variables whose distributions are known to the aggregation server and the users. The reconstruction of the community distribution can be done in a two-step process:

- Reconstruct the distribution of Ax by considering $Bn + C$ as noise, then compute the distribution of $\log(Ax)$.
- Because $\log(Ax) = \log(A) + log(x)$, we could reconstruct the distribution of $\log(x)$ using the distribution of $\log(Ax)$ found above and the distribution of $\log(A)$. Finally, compute the distribution of x from the distribution of $\log(x)$.

Note that the transformation of random variables by $\log$ and $\exp$ is trivial because both functions are monotonic. The reconstruction method used in each step is the same as the method used in the Traffic Analyzer application (Section 4.2). Figures 4.6(a) and 4.6(b) plot the original weight distribution and the reconstructed weight distribution using the above method, respectively. In this experiment, we use the same method described in the Traffic Analyzer application to generate a big community (500 users). For

simplicity, we choose $C = 0$. A and B are drawn from uniform distribution between 0 and 10. We observe from the figures that the reconstructed community distribution is very close to the real distribution.

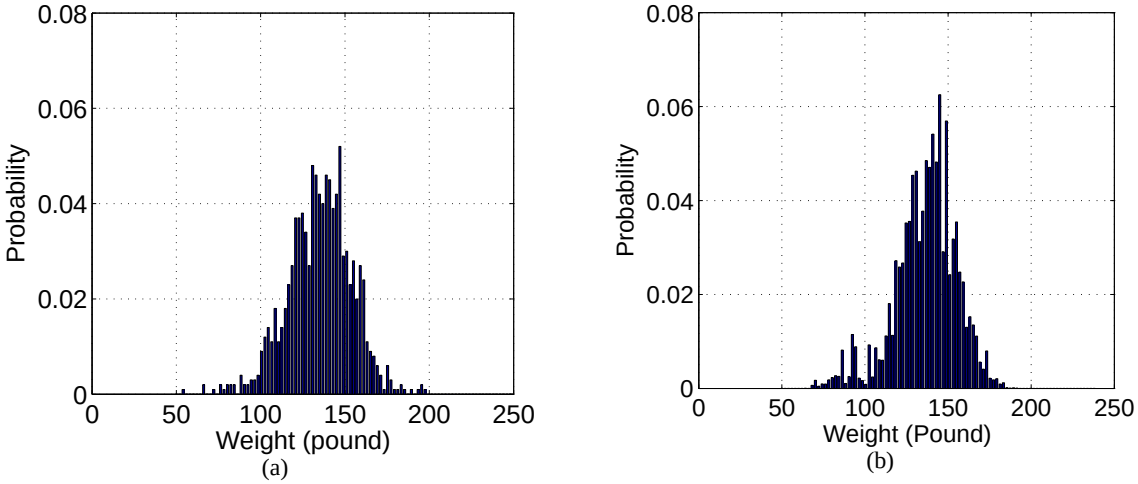


Figure 4.6: Figures showing real (a) and reconstructed (b) community weight distributions for one user

We observe from Figure 4.5 that the perturbed data contains a numbers of high frequency components, thus it is common to ask if the user data can be revealed using filtering techniques? We apply the PCA reconstruction method (same method used in the Traffic Analyzer application) to reconstruct an individual user's data. In order to employ PCA, we generated a virtual community containing 1000 users, where each user sends their perturbed data to the aggregation server. Figure 4.7 shows the real weight data, perturbed weight, and the reconstructed weight using PCA for a single user. The result shows that the reconstructed curve fits in the same direction as the perturbed data. Thus the filtering techniques again do not work with our perturbation scheme.

4.4 Conclusion

In this chapter, we presented a novel privacy preserving technique that utilizes data perturbation to achieve stream privacy. It ensures the privacy of individual user data against common reconstruction attacks and allows for the computation of statistics accurately. We demonstrated that the above algorithm can be utilized in a grassroots manner, one where there is no necessity for a trusted third party to compute the statistics. We also showed that the algorithm can be applied by non-expert users, individuals who are not experts at statistics or modeling. We demonstrated the application of the above algorithm in two different applica-

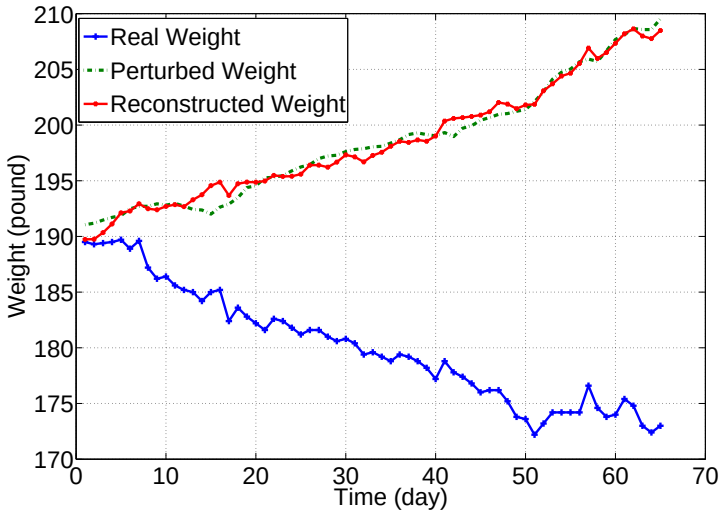


Figure 4.7: Graph showing the results of PCA reconstruction scheme on a single user

tions, Traffic Analyzer and Weight Watchers. These applications exemplify the privacy preserving nature of PoolView and the algorithm is implemented as part of the privacy firewall module of PoolView.

Chapter 5

Community Model Construction

Community sensing applications [22, 61, 76, 94] rely on individuals collecting sensor data and sharing it within a community (of common interest). These collected sensor data are then used for mapping common phenomena or computing community statistics. We discussed the computation of community statistics in the previous chapter (Chapter 4). Typically, large scale community sensing applications require a lot of data to compute or map the phenomena of interest. For example, in order to obtain a complete traffic map or an air quality map of a large city, individuals need to contribute a large number of traffic/air quality measurements within the area of interest. The challenge in such a scenario is to be able to generalize well from relatively sparse measurements of high-dimensional spaces to model the phenomena of interest. This is complicated by the fact that such phenomena are complex and trivial modeling techniques will fail to capture the entire phenomena.

In what follows, we will develop a novel GPS-based navigation service, called *GreenGPS*, that gives drivers the most fuel-efficient route for their vehicle as opposed to the shortest or fastest route. We will use GreenGPS to demonstrate a common problem in participatory sensing applications and present a solution methodology which can be extended to generic applications too. GreenGPS relies on data collected by individuals from their vehicles and a generalization framework that predicts the fuel consumption of an arbitrary car on an arbitrary street.

GreenGPS is possible thanks to the *On-Board Diagnostic* (OBD-II) interface, standardized in all vehicles that have been sold in the United States after 1996. The OBD-II is a diagnostic system that monitors the health of the automobile using sensors that measure approximately 100 different engine parameters. Examples of monitored measurements include fuel consumption, engine RPM, coolant temperature, vehicle speed, and engine idle time. A comprehensive list of measured parameters can be obtained from standard specifications as well as manufacturers of OBD-II scanners [9]. Several commercial OBD-II scanner tools

are available [3, 9, 10, 11], that can read and record these sensor values. Apart from such scanners, remote diagnostic systems such as GM's OnStar, BMW's ConnectedDrive, and Lexus Link are capable of monitoring the car's engine parameters from a remote location (e.g. home of driver of the car).

GreenGPS utilizes a vehicle's OBD-II system and a typical scanner tool in conjunction with PoolView's modules to enable data collection, upload, analysis, and mapping of fuel consumption data. In contrast to traditional mapping and navigation tools, such as Google maps [53] and MapQuest [72], which provide either the fastest or the shortest route between two points, GreenGPS collects the necessary information to compute and answer queries on the *most fuel-efficient route*. The most fuel-efficient route between two points may be different from the shortest and fastest routes. For example, a fastest route that uses a freeway may consume more fuel than the most fuel-efficient route because fuel consumption increases non-linearly with speed or because it is longer. Similarly, the shortest route that traverses busy city streets may be suboptimal because of downtown traffic.

The motivation for GreenGPS does not need elaboration. GreenGPS users might be driven by benefits such as savings on fuel or reducing CO_2 emissions and the carbon footprint. With the increase in the use of Bluetooth devices (e.g., cell-phones) and in-vehicle Wi-Fi, GreenGPS can be easily supported by inexpensive OBD-II-to-Bluetooth or OBD-II-to-WiFi adaptors that can upload OBD-II measurements opportunistically, for example, to applications running on the driver's cell phone [84]. It can also be supported by scanning tools that read and store OBD-II measurements on storage media such as SD cards. At the time of writing, OBD-II Bluetooth adaptors, such as the ELM327 Bluetooth OBD-II Wireless Transceiver Dongle, are available for approximately $50, together with software that interfaces them to phones and handhelds.

GreenGPS supports two types of users; members and non-members. Members are those who own OBD-II adaptors or scanning tools and contribute their data to the GreenGPS repository from the OBD-II sensors described above. They have GreenGPS accounts and benefit from more accurate estimates of route fuel-efficiency, customized to the performance of their individual vehicles.

Non-members can use GreenGPS to query for fuel-efficient routes as well. Since GreenGPS does not have measurements from their specific vehicles, it answers queries based on the average estimated performance for their vehicle's make, model, and year (or some subset thereof, as available).

We make two application specific contributions. First, we demonstrate how to use participatory sensing to develop a fuel-saving navigation service that relies on voluntary data collection by individuals to influence their routing decisions. Second, we provide a brief experimental evaluation of the system, where users are shown to save 6% in fuel on average over the shortest route and 13% in fuel over the fastest.

A general participatory sensing applications related contribution is to demonstrate how sparse samples of high-dimensional spaces can be generalized to develop models of complex nonlinear phenomena, where one size (i.e., model) does not fit all. We develop prediction models that enable us to extrapolate from fuel-efficiency data of some vehicles on some streets to the fuel consumption of arbitrary vehicles on arbitrary streets. While, in this case, the utility of such extrapolation may be short-term (soon all cars will be able to measure their own fuel-efficiency), the basic mechanisms and principles behind it can be used for a variety of other participatory sensing applications that share the need for generalizing from sparse data.

GreenGPS utilizes prediction models, developed in this chapter, to abstract vehicles and routes by a set of parameters such that fuel efficiency can be computed simply by plugging in the parameters of the right car and route. Using Dijkstra's algorithm, the minimum-fuel route can then be computed. An experimental study is performed over the course of three months using sixteen different cars with different drivers (and a total of over 1000 miles driven) to determine the accuracy of prediction models. It is shown that a prediction accuracy of 1% is attainable.

The rest of this chapter is divided into eight sections. Section 5.1 presents a feasibility study that investigates the amount of fuel savings that can be achieved by using GreenGPS and by following fuel-efficient routes. The details of GreenGPS system are described in Section 5.2. Models for estimating fuel consumption are presented in Section 5.3. Implementation details and evaluation results are presented in Section 5.5 and Section 5.6, respectively. Section 5.7 discusses the results and lessons learned. Finally, we conclude with directions for future work in Section 5.8.

5.1 A Feasibility Study

In this Section, we present a feasibility study that provides the reader with a proof of concept estimate of fuel savings that can be achieved by driving on the most fuel efficient routes.

We compute fuel consumption between landmarks in Urbana-Champaign for three different cars (and

drivers) and compare these values across multiple routes between the same pairs of landmarks. The landmarks chosen were frequently visited destinations such as the Siebel Center, Walmart, and the football stadium The shortest and fastest routes were obtained using MapQuest [72] [1]. In Figure 5.1, we plot the fuel consumption for the shortest route, the fastest route, and the route that consumes the least fuel (as computed from our models) for the aforementioned landmarks.

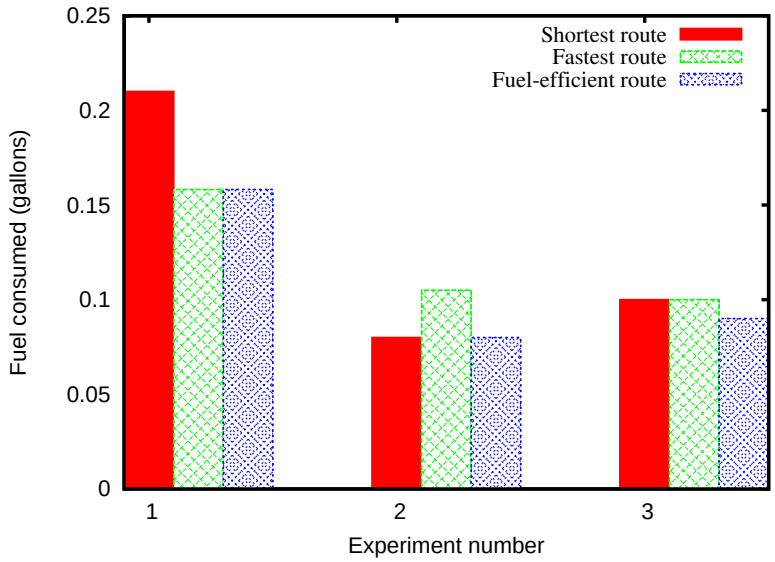


Figure 5.1: Figure showing fuel consumption for multiple routes between multiple selected landmarks for different cars and drivers

We observe from Figure 5.1, that in the first experiment, the fastest route is also the most fuel-efficient route. In the second experiment, the shortest route consumes the least amount of fuel. In the third experiment, the most fuel-efficient route is different from both the shortest and the fastest routes. We conclude from the above observations that simply choosing the shortest or the fastest route will not always be fuel-optimal.

For example, if the user always chooses the fastest route, their extra fuel consumption compared to taking the optimal route is 0%, 24%, and 10% for the three landmarks, respectively (an average of about 11% overhead). Similarly, if the user always chooses the shortest route, their average extra fuel consumption is about 11.5%. Hence, following the fuel-optimal route can translate (at the current national average gas

[1]Google maps provides only the shortest route. MapQuest provides both fastest and shortest routes. Hence, we use MapQuest to get route information.

price, which at the time of writing this paper was USD 2.86 [1]) into savings of at least 30 cents per gallon, which is not bad for "cash back".

The above results are only a proof of concept. They simply show that there may exist situations where using a fuel-optimal route can save money. A more extensive study of route models and savings is presented in the evaluation section.

To estimate the amount of savings that can be achieved on a global scale, we provide back of the envelope calculations based on data from the Environmental Protection Agency (EPA) [36]. An estimated 200 million light vehicles (passenger cars and light trucks) are on the road in the US. Each of them is driven, on an average, 12000 miles in a year. The average mile-per-gallon (mpg) rating for light vehicles is 20.3 mpg. Even if 5% of these vehicles adopted GreenGPS and 10% fuel savings were achieved on only a quarter of the routes traveled by each of these vehicles, the amount of overall fuel savings is nearly 177 million gallons of fuel ($(12000/20.3) * 0.3 * (0.05 * 200M) * 0.1$). This translates into nearly half a billion dollars in savings at the pump (based on the current national average pump prices for a gallon of gasoline). We consider the above prospective savings acceptable.

5.2 The GreenGPS System

The service provided by GreenGPS is similar to a regular map application, such as Google maps [53] or MapQuest [72]. Google maps and MapQuest provide the shortest or fastest routes between two points, whereas GreenGPS computes the most fuel-efficient route. A snapshot of the Web-based GreenGPS's user interface, which was created using PoolView is shown in Figure 5.2 along with the most fuel efficient route between two points for a user with a Toyota Celica, 2001. In the following subsections, we will discuss the GreenGPS concept followed by how individuals use PoolView for data collection and data sharing and the specifics of the hardware used for the purpose of data collection.

5.2.1 The GreenGPS Concept

Individuals who want to compute the most fuel-efficient route between two points enter the source and destination address via the interface provided by GreenGPS. Members of GreenGPS (i.e., those individuals who contributed participatory data) can register their vehicles that were used for data collection. Hence,

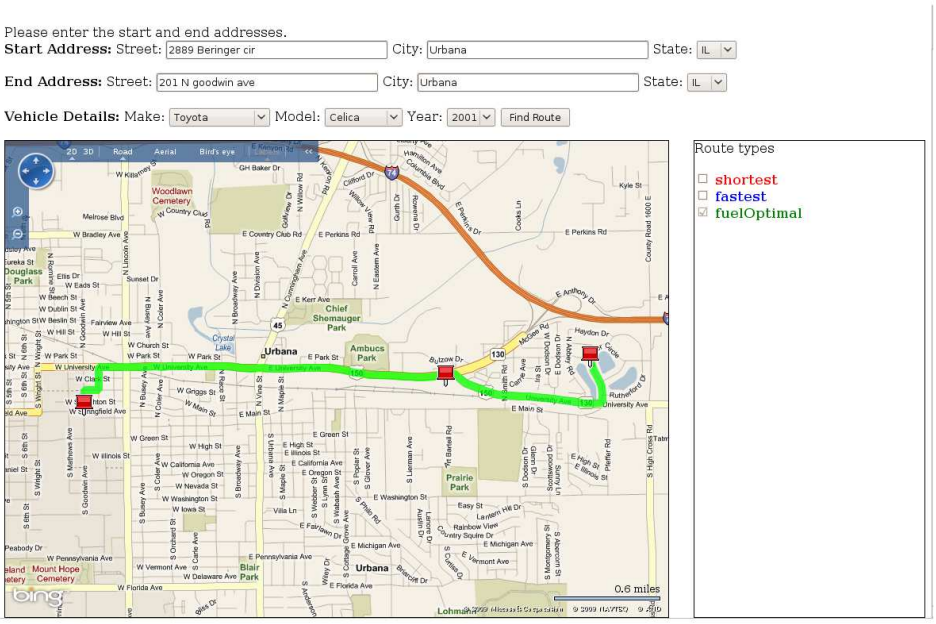


Figure 5.2: Figure showing the user interface of GreenGPS with the most fuel efficient route between two points on the map for a Toyota Celica, 2001.

GreenGPS can compute the route specifically for the registered vehicle. Other users may enter their vehicle's make, model, and year of manufacture. Since different vehicles have different fuel consumption characteristics, these car details are used to compute the most fuel-efficient route for the given vehicle brand. The advantage for the users who contribute data is that the system provides better estimates of the most fuel-efficient routes to these individuals, thus allowing them to have higher savings.

Currently, it is impractical to assume that GreenGPS members will measure all city streets and cover all vehicle types. Instead, measurements of GreenGPS members are used to calibrate generalized fuel-efficiency *prediction models*. These models, discussed in Section 5.3, show that the fuel consumption on an arbitrary street can be predicted accurately from a set of *static* street parameters (e.g., the number of traffic lights and the number of stop signs) and a set of *dynamic* street parameters (such as the average speed on the street or the average congestion level), plus of course the vehicle parameters (e.g., weight and frontal area). It is the mathematical model describing the relation between these general parameters and fuel-efficiency that gets estimated from participant data. Hence, the larger and more diverse is the set of participants, the better the generalized model.

For most streets, static street parameters can be readily obtained from traffic databases. For example,

the number of traffic lights and the number of stop signs on streets can be obtained from the red light database [54]. Dynamically changing parameters such as the congestion levels or average speed are more tricky to obtain. In larger cities, real-time traffic monitoring services can supply these parameters [101]. Many GPS device vendors, such as TomTom, also collect and provide congestion information. Finally, our participatory sensing service, Traffic Analyzer, developed earlier in Chapter 4 has the potential to provide congestion and speed data.

In our deployment, speed information is obtained from the collected data using the hardware described in the next section. The speed data is aggregated for different city blocks, based on the GPS location information. Thus, given a street name (or the latitude/longitude of a location), GreenGPS provides the average speed information for the corresponding block.

5.2.2 GreenGPS Implementation using PoolView

Our architecture, PoolView, is used to implement GreenGPS. It facilitates the storage and sharing of OBD-II sensor data. We implemented GreenGPS by writing our aggregation server for PoolView. An individual who wants to share their OBD-II sensor data uses PoolView's client side interface to upload their data to the GreenGPS aggregation server. The aggregation server uses these data to calibrate models that relate street and vehicle parameters to fuel-efficiency and offers the GreenGPS query interface for fuel-efficient routes.

Individuals who wish to contribute OBD-II data to GreenGPS install, in their vehicle, a commercial OBD-II scanner along with a GPS unit. In our deployment, we use one such off-the-shelf device for data collection purposes, called DashDyno [9], shown in Figure 5.3. The DashDyno's OBD-II scanner is interfaced to a Garmin eTrex Legend GPS [50] to get location data. The DashDyno records trip data (including Garmin's GPS location) on an SD card that the user later uploads it to the GreenGPS server (using PoolView).

A total of 16 parameters are obtained from the car and the GPS, the most important of them being instantaneous vehicle speed, total fuel consumption, rate of fuel consumption, latitude, longitude, and time.

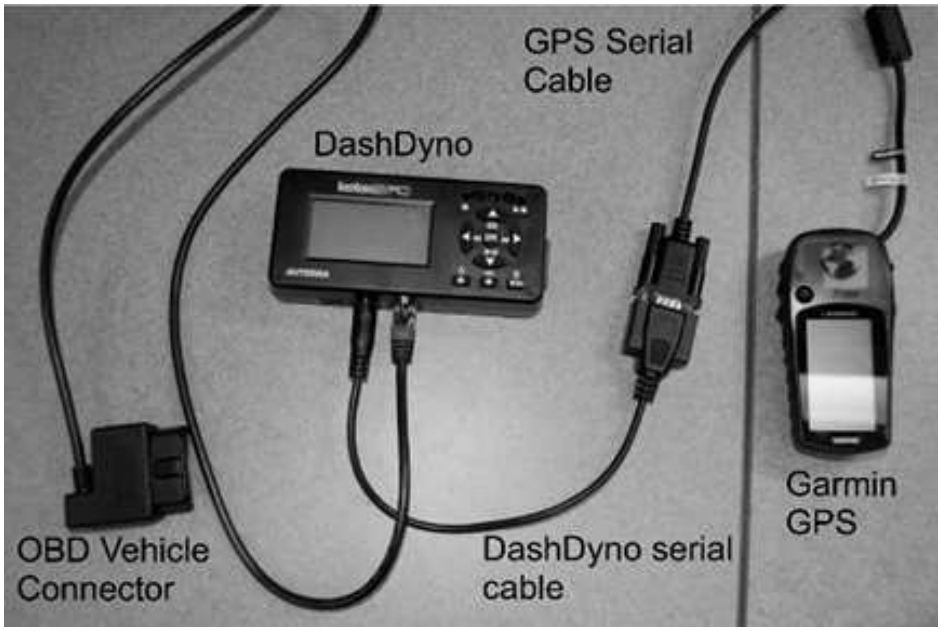


Figure 5.3: Hardware used for data collection

5.3 Generalizing from Sparse Data

In this section, we demonstrate the foundations of one of the key mechanisms in participatory sensing applications that are tolerant to conditions of sparse deployment; namely, the generalization from sparse multidimensional data. Such generalization is complicated by the fact that, in high-dimensional data sets, one size does not fit all. Hence, for example, developing a single regression model to represent all data is highly suboptimal. In the case of GreenGPS, the lack of widespread availability of OBD-II scanner tools suggests that the data contributed by users of our participatory sensing application will be a sparse sampling of routes and cars. Hence, we aim to use data collected by a smaller population to build models capable of predicting the fuel consumption characteristics of a larger population. Admittedly, the conditions of sparse deployment are typically temporary (in the case of GreenGPS), making the above contribution short-lived in nature. Nevertheless, it solves a key problem at a critical phase of most newly deployed systems, which makes it important. Before we explain the details of the generalization mechanism, we will provide a brief description of our data collection for the purpose of developing models.

5.3.1 Data Collection

The vision for GreenGPS, when fully deployed, is to collect data from everyday users, which can be employed to update and refine predictions of fuel consumption when such users (or others with similar vehicles) embark on new itineraries. In this chapter, we conducted a limited proof-of-concept study involving sixteen users (with different cars) over the course of three months. A total of over 1000 miles were driven by our users to construct the initial models. Figure 5.4 shows a partial map of the paths on which data were collected. The details of the car make, model, year, and the number of miles of data collected for each car are summarized in Table 5.1.

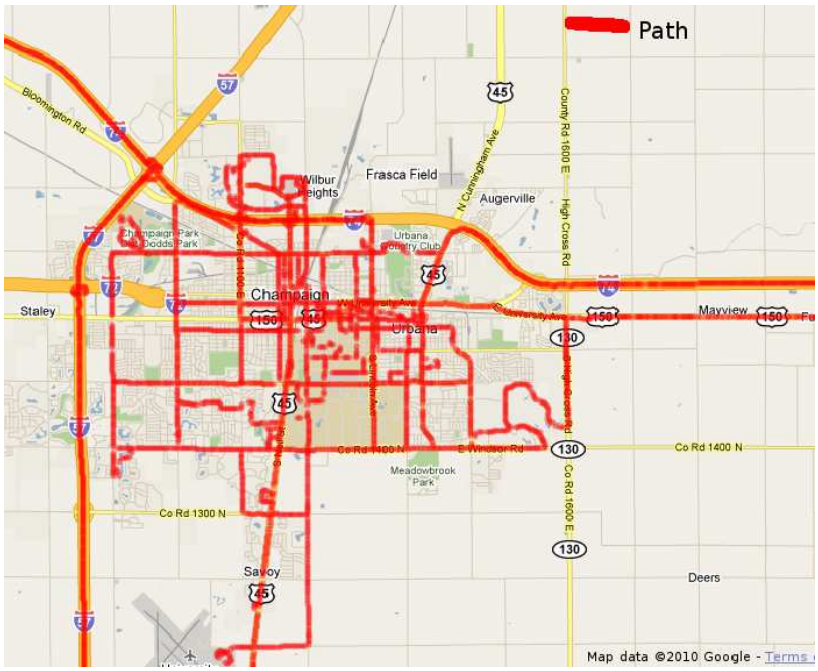


Figure 5.4: Coverage map for the paths on which data were collected

In the aforementioned experiments, each user was asked to drive among a specific set of landmarks in the city. We split each drive into smaller *segments* to capture the variation in the fuel consumption on individual streets. These segments were the "samples" used to capture the variables affecting fuel consumption and develop initial prediction models.

Car make	Car model	Car year	Miles driven
Ford	Taurus	2001	135
Toyota	Solara	2001	45
BMW	325i	2006	70
Toyota	Prius	2004	140
Ford	Taurus	2001	136
Ford	Focus	2009	95
Toyota	Corolla	2009	45
Honda	Accord	2003	102
Ford	Contour	1999	22
Honda	Accord	2001	18
Pontiac	Grand Prix	1997	25
Honda	Civic	2002	11
Chevrolet	Prizm	1998	16
Ford	Taurus	2001	10
Mazda	626	2001	9
Toyota	Celica	2001	120
Hyundai	Santa Fe	2008	22

Table 5.1: Table summarizing the cars used and the amount of data collected

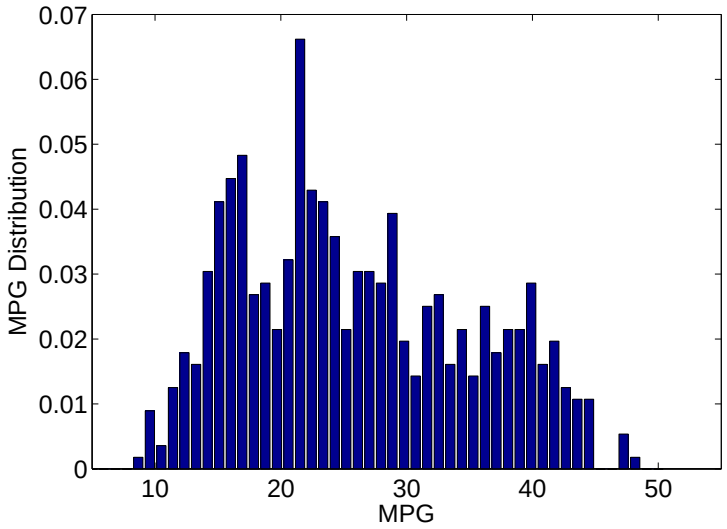


Figure 5.5: Figure showing the real mpg distribution for all the sixteen users

5.3.2 Derivation of Model Structure

The first part of data generalization is to derive a model structure. The structure describes how various parameters are related, but does not evaluate the various constants and proportionality coefficients. In this

case, we derive the structure of fuel prediction models from physical analysis.

To motivate the need for modeling, we plot the distribution of miles per gallon (mpg) for all the data collected in Figure 5.5. We observe from this figure that the distribution is nearly uniform with the mpg values varying between 5 and 50. The standard deviation of the mpg distribution is 9.12 mpg, which is pretty high. Hence, an appropriate model is needed to estimate the fuel consumption on various segments.

The inputs to our prediction model include segment parameters and car parameters. We do not consider driver factors in the model because the sample size of drivers was small in our dataset. Note that, we are interested in predicting long-term fuel consumption only. While actual savings of a user on individual commutes to work may vary, the user might be more concerned with their net long-term savings. Hence, it is important to capture only the statistical averages of fuel consumption. As long as the errors have near zero mean, the savings are accurate in the long term. As a given user drives more segments, a value of interest is the total end-to-end prediction error that results (which improves over time as the individual positive and negative segment errors cancel out). We call that end-to-end error the *cumulative error*. It is useful to normalize that error to the total distance driven. We call the result *cumulative percentage error*. It represents how far we are off in our estimate of total fuel consumption.

5.4 Model Structure Derivation

We will derive the model structure for fuel consumption from the basic principles of physics. Many such models exist in prior literature [20, 40, 102], which simplifies the task. We divide the parameters that affect fuel consumption into (i) *static segment parameters*, namely, numbers of stop signs (ST), numbers of traffic lights (TL), distance traveled (Δd) and slope (θ), (ii) *dynamic segment parameters*, namely, average speed ($\bar{v}$), and *car specific parameters*, namely, weight of the car (m) and car frontal area (A).

Assuming that the engine RPM is ωs^{-1}, the torque generated by the engine is $\Gamma(\omega)$, the final drive ratio is G, the $k-th$ gear ratio is g_k, and the radius of the tire is r, then F_{engine} is given by the following equation:

$$F_{engine} = \frac{\Gamma(\omega) G g_k}{r} \tag{5.1}$$

The frictional force $F_{friction}$ is characterized by the gravitational force acting on the car, given by

$mgcos(\theta)$, where m is the mass of the vehicle and g is the gravitational acceleration and the coefficient of friction, c_{rr}. The equation for frictional force is:

$$F_{friction} = c_{rr} mgcos(\theta) \tag{5.2}$$

The gravitational force, F_g, due to the slope is given by the following equation:

$$F_g = mgsin(\theta) \tag{5.3}$$

Finally, the force due to air resistance, F_{air}, is given by the following equation:

$$F_{air} = \frac{1}{2} c_d A \rho v^2 \tag{5.4}$$

In the above equation, c_d is the coefficient of air resistance, A is the frontal area of the car, ρ is the air density, and v is the current speed of the car.

Assuming that the car is on an upslope, the final force acting on the car is given by the following equation:

$$F_{car} = F_{engine} - F_{friction} - F_{air} - F_g \tag{5.5}$$

In order to obtain a relation between the fuel consumed and the above forces, we note that the fuel consumed is related to the power generated by the engine at any instance of time, t. If f_r is the fuel rate (fuel consumption at a given time instance) and P is the instantaneous power, then $f_r \propto P$. Power is related to the torque function, $\Gamma(\omega)$, and engine RPM, ω as follows: $P = 2\pi\Gamma(\omega)\omega$. Hence, we obtain $f_r = \beta\Gamma(\omega)\omega$.

In the above equation, β is a constant. Further, we also have the relationship $v = r\omega$ from rotational dynamics. From the above equations, we obtain the fuel consumption rate as a function of the forces acting

on the car shown below:

$$\begin{aligned} F_{car} &= ma \\ &= \frac{f_r G g_k}{r\omega\beta} - c_{rr} mgcos(\theta) - \frac{1}{2} c_d A\rho v^2 - mgsin(\theta) \quad (5.6) \\ mav &= \beta' f_r - c_{rr} mgcos(\theta) v - \frac{1}{2} c_d A\rho v^3 - mgvsin(\theta) \quad (5.7) \\ f_r &= k_1 mav + k_2 mvcos(\theta) + k_3 Av^3 + k_4 mvsin(\theta) \quad (5.8) \end{aligned}$$

Finally, we can obtain the equation for the fuel consumed, f_c by integrating the rate of fuel consumption with respect to time. We obtain the following equation:

$$\begin{aligned} f_c &= \int_{t_1}^{t_2} f_r(t)\, dt \\ &= \int_{t_1}^{t_2} (k_1 mav + k_2 mvcos(\theta) + k_3 Av^3 + k_4 mvsin(\theta))\, dt \quad (5.9) \end{aligned}$$

In order to further derive a model that can be used for regression analysis, we will detail the various components that are part of the fuel consumption of a car. As shown in the above equation, a moving car at a constant speed on a straight road which does not encounter any stop lights or traffic will only need to overcome the frictional forces caused by the road, the air, and gravity. These are represented by $\int_{t_1}^{t_2} k_2 mvcos(\theta)$, $\int_{t_1}^{t_2} k_3 Av^3$, and $\int_{t1}^{t2} k_4 mvsin(\theta)$, respectively. On the other hand, the first component $\int_{t_1}^{t_2} k_1 mav$ can be broken down further into two components, one is the extra fuel consumed due to encountering stop signs (ST) and traffic lights (TL) and the second one is the extra fuel consumed due to congestion. Hence, the previous equation now becomes the following:

$$f_c = \int_{t_1}^{t_2} (k_{11} mav(ST + \nu TL) + k_{12} mav) + k_2 mvcos(\theta) + k_3 Av^3 + k_4 mvsin(\theta))\, dt \quad (5.10)$$

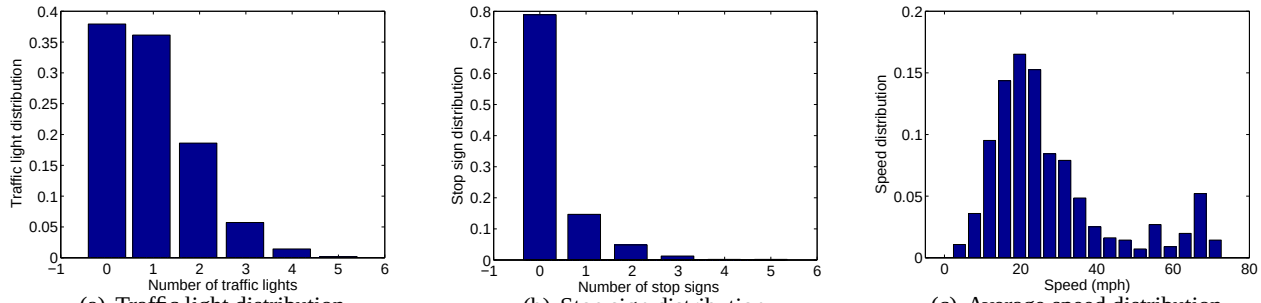
If we replace v with $\bar{v}$, the average speed, assume that θ remains constant, and we know $a = dv/dt$, we can further simplify the above integral to the following:

$$f_c = k_{11} m\bar{v}^2 (ST + \nu TL) + \frac{k_{12} m\bar{v}^2}{2} + k_2 m\Delta dcos(\theta) + k_3 A\bar{v}^3 \Delta t + k_4 m\Delta dsin(\theta) \quad (5.11)$$

In the above equation, Δd is the distance traveled and Δt is the time traveled. Dividing the above equation by Δd gives us the metric *fuel consumed per mile (gpm)*, which is appropriate for our analysis purposes. Hence, our final model will now be (replacing the constants by k_1, k_2, k_3, k_4 and k_5:

$$gpm \quad = \quad k_1 m\bar{v}^2 \frac{(ST + \nu TL)}{\Delta d} + k_2 m \frac{\bar{v}^2}{2\Delta d} + k_3 mcos(\theta) + k_4 A\bar{v}^2 + k_5 msin(\theta) \tag{5.12}$$

We plot the distributions for various parameters (for individual segments) in Equation 5.12 for the data that we collected in Figure 5.6. In the next section, we show that the coefficients of our model, k_1, k_2, k_3, k_4



(a) Traffic light distribution (b) Stop sign distribution (c) Average speed distribution

Figure 5.6: Figures showing the distributions of number of traffic lights, stop signs, and average speed

and k_5 differ among different vehicles making it harder to generalize from vehicles we have data for to those we do not.

5.4.1 Model Evaluation: One Size Fits All?

Regression analysis is a standard technique for estimating coefficients of models with known structure. In this section, we demonstrate that a single regression model is a bad fit for our data. Said differently, while a regression model that accurately predicts fuel consumption can be found for each car from data of that one car, the model found from the collective data pool of all cars is not a good predictor for any single vehicle. Hence, in a sparse data set (where data is not available for all cars); it is not trivial to generalize. We illustrate that challenge by first evaluating the performance of car models obtained from their own data (which is good), then comparing it to the trivial generalization approach: one that finds a single model based on all car data then uses it to predict fuel consumption of other cars. A solution to the challenge is presented in the next section.

One should add that while the generalization challenge is common to many participatory sensing applications, our evaluation is not intended to be a definitive study on vehicular fuel consumption. For example, we evaluate fuel consumption in Urbana-Champaign only, which is quite flat. Hence, $\theta = 0$ is a good approximation. (We therefore set the last term, $k_5 msin(\theta)$, of our physical model to zero, so k_5 is no longer needed.) Furthermore, the city is rarely congested. Moreover, the range of cars used in the study is rather skewed towards sedans, and hence not representative of the diversity of cars on the streets. Fortunately, even this rather homogeneous data set is sufficient to show that generalization is hard.

First, we determine the length of the segment empirically. We vary the segment length from 0.5 miles to 2 miles in increments of 0.5 miles and evaluate the accuracy of our model in each of these cases. We observed that the accuracy of the model is best when the segment length is 1 mile. Hence, we fix the segment length to be 1 mile in the rest of our experiments. We evaluate the accuracy of models derived from vehicle data using a cross validation approach. We choose a random data point (i.e., a given *segment* of a street driven by some car) to predict fuel consumption for. We then use other points to train a model. We distinguish models based on other segments of the same car from models based on data from other cars in predicting the fuel consumption of the one segment. The 4th and 5th columns of Table 5.2 summarize the resulting errors, respectively, for a fraction of the used cars.

Car make	**Car model**	**Car year**	**Ind. cumulative error%**	**General cumulative error%**
Hyundai	Santa Fe	2008	2.89	23.63
Honda	Accord	2003	0.89	15.3
Ford	Contour	1999	0.83	91.4
Ford	Focus	2009	0.12	27.35
Ford	Taurus	2001	0.75	24.85
Toyota	Corolla	2009	0.61	89.97
Ford	Taurus	2001	0.56	6.9

Table 5.2: Table summarizing the cumulative percentage errors for the individual car models and the generalized case when all the data (except one car) is used to obtain the model

We also plot the error distribution for individual segments (for one car) in Figure 5.7. We observe that this distribution is near normal and the mean is near zero (0.26%). We observe a similar distribution for other cars too.

We also observe from the Table 5.2 that the cumulative percentage error for individual car models are

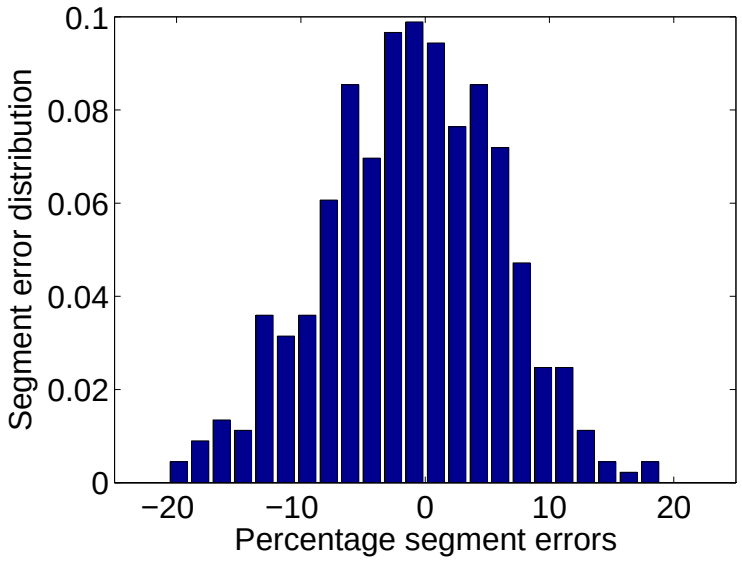


Figure 5.7: Figure showing the segment error distribution for one car

quite good. Most of them are below 2%. On the other hand, when we predict one car's consumption using data from other cars, the errors are quite high. This suggests the existence of non-trivial bias in error that does not cancel out by aggregation. In the next section, we propose a way to mitigate this problem based on grouping cars into clusters, such that prediction can be done based on other *similar* cars by some metric of similarity.

5.4.2 Model Clustering

The above suggests a need for better generalization over vehicle data. Different car types behave differently. Even though the model is parameterized by factors such as car weight and frontal area, they are not enough to account for differences among cars. This is a common problem in high-dimensional data sets collected in participatory sensing applications. The question becomes, if we cannot generalize over the whole set, can we generalize over a subset of dimensions?

We propose a solution that utilizes a popular approach from data mining literature, *data cubes* [55]. Data cubes are structures for Online Analytical Processing (OLAP) that are widely used for multidimensional data analysis. They group data using multiple attributes and extract similarities within each group. For example, previous work showed how to efficiently construct regression models for various subsets of

data [26]. The data cube framework can thus help compute the optimal generalization hierarchy in that it can help generalize data based on those dimensions that results in the minimum modeling error.

We consider three major attributes (data dimensions) of a given car: *make*, *year*, and *class*. Based on these three attributes, data can be grouped in eight ways. At one extreme, all cars may be grouped together, thus producing a single regression model (which we have shown is not acceptable). At the other extreme, cars can be partitioned into clusters based on their (make, year, class) tuple. A separate model is derived for each cluster. Therefore, a 2001 compact Ford is modeled differently from a 2001 mid-size Ford, a 2002 compact Ford or a 2001 compact Toyota.

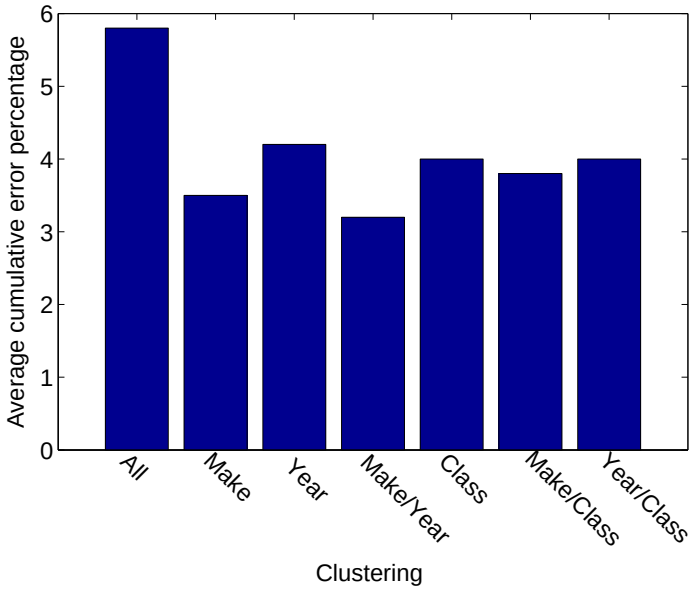


Figure 5.8: Cumulative error percentage of the models obtained from various clustering approaches

Between these two extremes, to find out which clustering scheme gives the best accuracy, we obtain the cumulative percentage error for each scheme. The results, summarized in Figure 5.8, show that different generalizations have different quality. These generalizations are somewhat better than using all car data lumped together. While our data set is too small to make general conclusions (from only 16 cars), as more data are collected in a deployed participatory sensing application (e.g., say deployment reaches 100s of cars), progressively better generalizations can be attained.

To use results of Figure 5.8, one would build models for each pair of make and year (the lowest error

clustering scheme). If a car is encountered for which we do not have data on its (make, year) cluster, we go one level up and use (make) clusters or (year) clusters as generalizations for the (make, year) cluster. If there are no models corresponding to either make or year of a given car, we have no recourse but to go one level up and use the model computed from all data. Figure 5.9 depicts the generalization process among various model clusters.

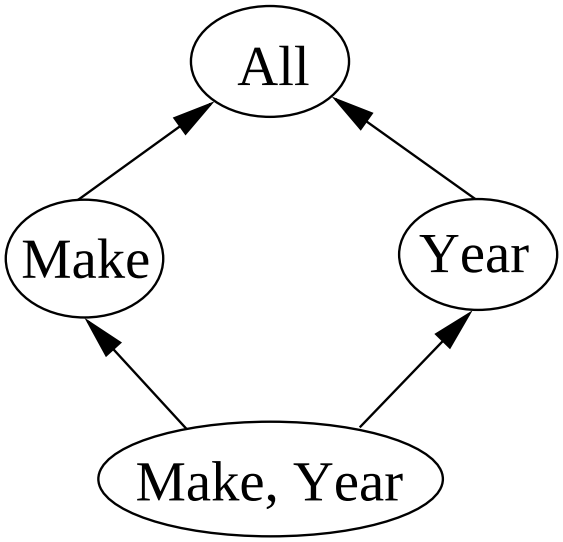


Figure 5.9: Model generalization from fine grained clusters

Car make	Car model	Car year	Cumulative error %
Hyundai	Santa Fe	2008	0.73
Honda	Accord	2003	1.01
Ford	Contour	1999	1.42
Ford	Focus	2009	2.7
Ford	Taurus	2001	3.38
Toyota	Corolla	2009	1.28

Table 5.3: Table showing the cumulative error percentage for each individual car when model clustering is used

We evaluate the performance of our model clustering technique by measuring how accurately an individual car can be modeled using the data from cars with similar make or year. Specifically, we construct the model cluster while removing data of a certain car type. We use the model cluster to estimate the fuel consumption for a given car. The resulting cumulative error percentage is presented in Table 5.3.

To put the above results in perspective, the reader is reminded that the nature of the landscape in Urbana-

Champaign limits our study in that we do not have data on hilly terrain. The study would have been more interesting if conducted on uneven grounds, where changes in incline modulate fuel consumption. We expect that future data collected will be used to evolve our current model by considering the terrain (θ in Equation 5.12) parameter. Further, new data collected will be used to update the model. Another limitation of our modeling approach arises from the class of cars for which data has been collected. We observe from Table 5.1 that the majority of the cars are sedans (with the exception of one SUV). We observe that the generalization tree (Figure 5.9) does not use the *class* of the car. This generalization tree is specific to the dataset collected. The point of this section is to illustrate an approach to improve prediction in the temporary but important conditions of sparse deployment. Ultimately, when all cars have their own OBD-II readers supplying data to drivers' cell-phones, we shall not need the generalization scheme described above.

5.5 Implementing GreenGPS

The GreenGPS server is implemented as an aggregation server in PoolView (data is collected by the server from individual users is described in Section 5.2). The server utilizes various PoolView modules, which include the storage (server), community model construction, and map based application support tools.

GreenGPS uses an instance of the map-based application support tools to obtain the fuel efficient routes. It maintains the street variables affecting fuel consumption as additional parameters in the OSM map. This information is stored as a tag/value pair in the way object (of the OSM map), where tag is a street parameter and value is the corresponding value of the parameter. Further, the car parameters are maintained in a separate database, stored in the data storage component of the GreenGPS aggregation server. The model to compute fuel consumption on a given way (for a given car and driver) queries these databases and computes the fuel consumption on the way.

The OBD-II data shared by individuals is used to compute regression models that predict the fuel consumption on specific streets given the car details (e.g. make, model, age). The regression variables which are street specific are stored in the OSM map database, whereas the car specific variables are stored in a similar database.

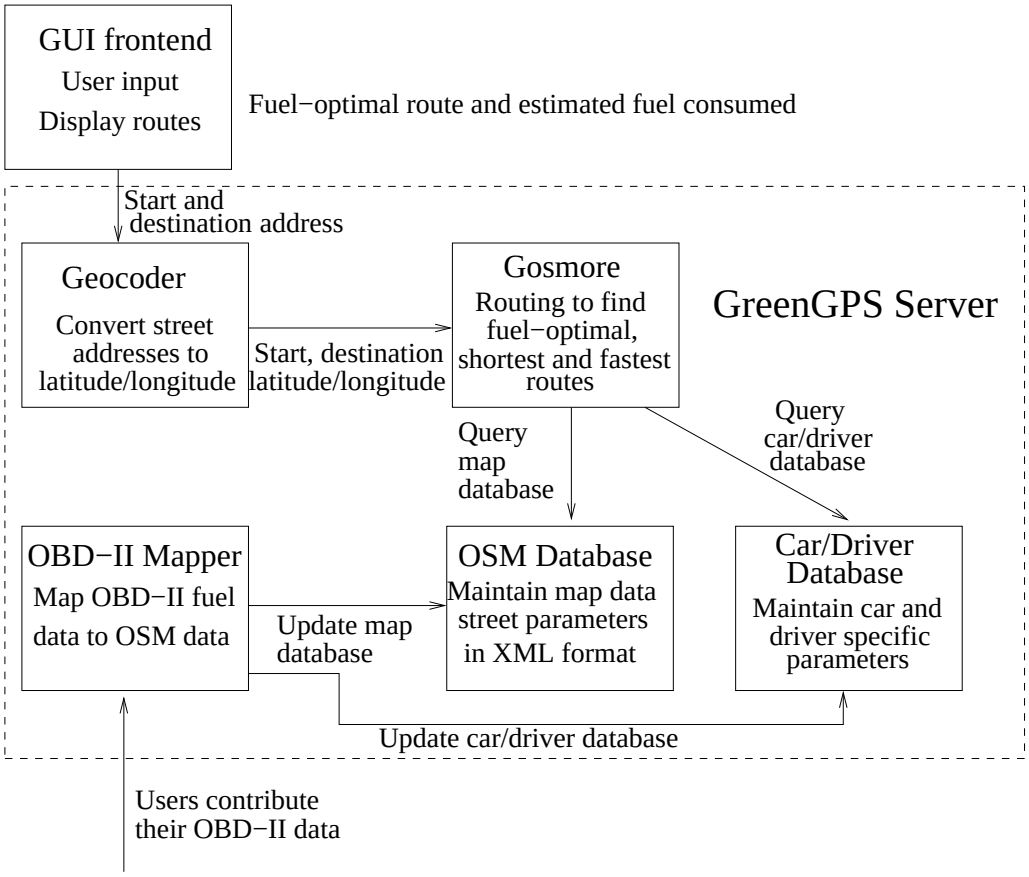


Figure 5.10: Figure depicting the various modules of GreenGPS

5.5.1 Model Clustering Implementation

GreenGPS model clustering is implemented as an instance of the community data modeling module of PoolView. First, this module takes as input the model structure (as derived in Section 5.3), the clustering parameters, and the collected data and creates the generalization hierarchy. Once the hierarchy has been established, this module can then be queried by providing as input the model structure parameters (such as number of stop signs, traffic signals, average speed, and car weight). The output of such a query will be the fuel consumed by the given car on the queried road segment.

5.5.2 Routing in GreenGPS

Routing is achieved in GreenGPS by customizing the open source routing software, Gosmore [80]. Gosmore is a C++ based implementation of a generic routing algorithm that provides shortest and fastest routes

between two arbitrary end-points. Gosmore uses OSM XML map data for doing routing. Gosmore's routing algorithm is a heuristic that by default computes the shortest route. This routing algorithm can be thought of as a weighted Dijkstra's algorithm on the OSM map, where the nodes of the graph are OSM nodes and the edges of the graph are OSM ways and the weights of the edges are the lengths (distance) of the ways. The fastest route is computed by multiplying the distance by an inverse speed factor (thus giving lower weights to faster ways). Our fuel-optimal routing algorithm multiplies the distance by an inverse mpg (miles per gallon) metric that results in lower weights for fuel-optimal ways.

5.5.3 Other Implementation Issues

Street address inputs provided by the user are translated into latitude/longitude pairs using PoolView's geocoding module.

The GUI frontend to display the fuel-optimal route (shown in Figure 5.2) utilizes Microsoft Bing maps. Routes are color coded and rendered as *polylines* on Bing maps. For example, the fuel-optimal route is a "green" color polyline.

When a query is posed to GreenGPS for the fuel-optimal route between the start address and destination address, the addresses are first geocoded into their corresponding latitude and longitude pairs using the geocoder module. The latitude and longitude pairs of the start and destination addresses are then fed to the routing module which computes the fuel-optimal route (along with the shortest and fastest routes) using the OSM XML database and the prediction models of fuel consumption on streets (computed from the OBD-II sensor data contributed by users). The computed routes are then displayed on the Bing maps based GUI frontend.

5.6 Evaluation

We evaluate the performance of GreenGPS in two stages. First, we evaluate the performance of our model by using it to predict the end-to-end fuel consumption for long routes. Second, we evaluate the potential fuel savings of an individual using GreenGPS.

5.6.1 Model Accuracy

We first evaluate the accuracy of our prediction model in estimating fuel consumption on long routes. These routes are continuous sequences of segments that individuals drove. Only six cars are used in this experiment [2] because the data from the rest of the cars did not include multiple paths (and hence we would not be able to do path-based cross validation, where data collected on one path is used to predict fuel consumption on another). We consider the path error as the end-to-end prediction error for the given path (which is the metric used for evaluation in Section 5.3). For cross validation, we remove the data points associated with a given path and obtain a model for the car, then obtain the error in predicting fuel consumption for this path based on the computed model. We repeat the above for all the paths.

The entire path error distribution corresponding to the above experiment when prediction for each car is used based on data of the same car (on other paths) is shown in Figure 5.11. We observe that the path error distribution is nearly normal and that the mean of this distribution is near zero ($<$1%). We conduct a similar experiment to derive the path error distribution that is achieved by employing clustering such that fuel consumption of cars is predicted from that of other cars in the nearest cluster. To experiment with prediction accuracy of clusters, we remove the data points for each car (as if that car was not known to the system) and cluster the rest of data points, as described in Section 5.4.2, based on make, year, and both. Fuel consumption of the removed car is then predicted using the nearest cluster. Namely, we first check if a cluster exists with the same car make and year. If no such cluster exists, we check for a cluster of the same make or the same year, respectively. Finally, a model based on all car data is used if all the previous steps fail. The prediction error for each path is computed as before and the distribution is presented in Figure 5.12. Again, a normal distribution of the path errors is observed with near zero mean ($<$4%).

In order to understand how path errors vary with path lengths, we bin the paths based on their length and compute the average of the absolute path errors as a function of path length. We repeat this experiment for the case where models are derived for each car individually and the case where models are derived for clusters and the nearest cluster is used. We plot the mean of the absolute path errors for varying path lengths in Figure 5.13.

We observe from Figure 5.13 that the error decreases with increasing path length for both the individual

[2] Ford Focus, 2009; Ford Taurus 2001; Toyota Corolla, 2009; Ford Taurus, 2001, Honda Accord, 2001; and Ford Taurus, 2001.

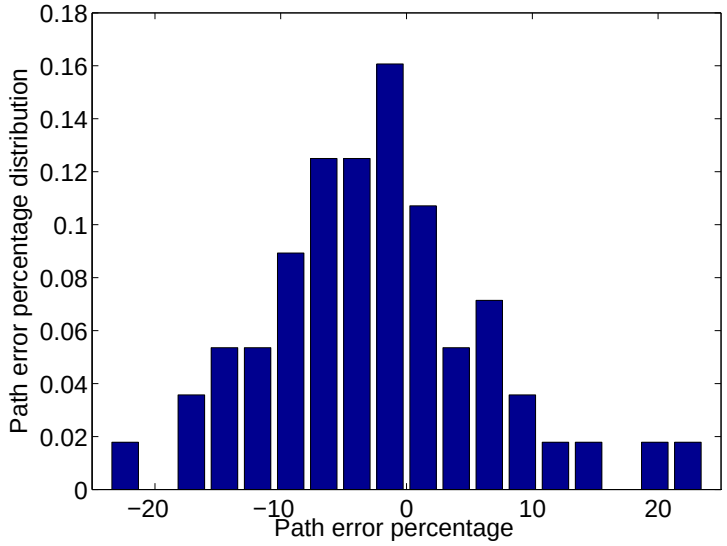


Figure 5.11: Distribution of path error percentages when training is done using individual cars

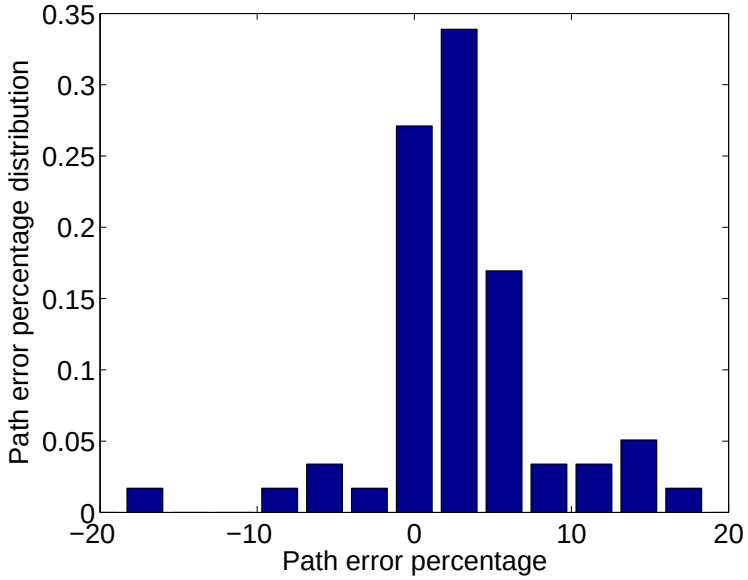


Figure 5.12: Distribution of path error percentages for the clustering approach

and cluster based models. As expected, models based on the owner's car do better than models based on the nearest cluster, but the cumulative error continues to decrease with distance driven, which is what we want. We have not explored if this holds true when the commutes have large dynamics in speeds, such as in larger cities. The current data set is limited in that it was collected in a fairly quiet town.

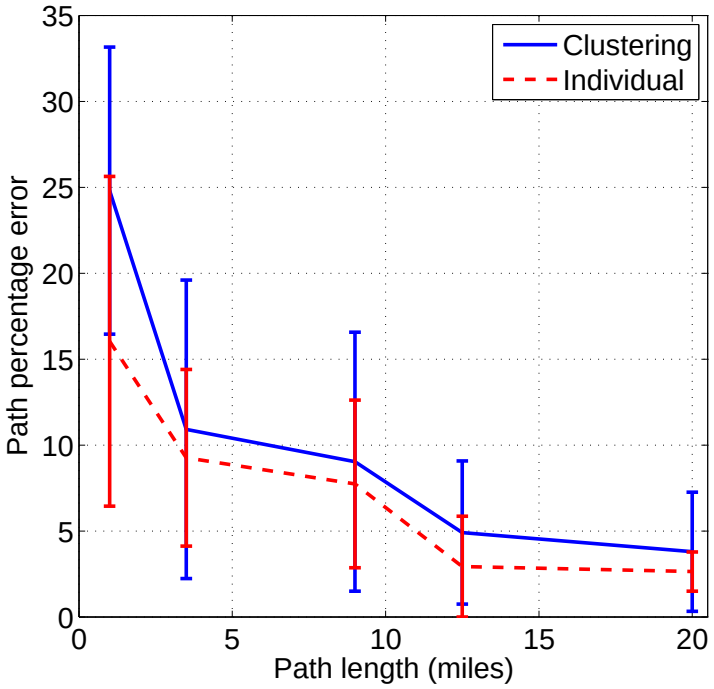


Figure 5.13: Mean path error when path length is varied for individual car models and cluster based models

From the perspective of building participatory sensing applications, the above suggests the importance of finding models that do not have *biased error*. Since the models often try to predict aggregate or long-term behavior (such as long term exposure to pollutants, annual cost of energy consumption, eventual weight-loss on a given diet, etc), if the error in day-by-day predictions is normally distributed with zero mean, the long-term estimates will remain accurate. Hence, rather than worrying about exact models, GreenGPS attempts to find *unbiased* models, which is easier.

5.6.2 Fuel Savings

In this section, we evaluate the fuel savings achieved when using the GreenGPS system. To evaluate fuel savings, we chose landmarks in the city of Urbana-Champaign that are visited by the drivers in our study from their daily commutes, such as work, gym, frequently visited restaurants, and shopping complexes. To eliminate subjective choice of routes between the selected landmarks, we selected a pair of landmarks then looked up both the shortest route and fastest route between these landmarks on MapQuest. The person then drove eight round trips (of approximately 20-40 minutes each) between their selected pair of landmarks;

four on the shortest route and four on the fastest route, recording actual fuel consumption for each round trip. The landmarks together with the shortest and fastest routes are shown in Figure 5.14. We then used the GreenGPS system to predict which of the two compared routes for each pair of landmarks is the better route, which it did correctly in every case.

The fuel consumption data for each roundtrip on the shortest and fastest routes for all the cars in this experiment are shown in Table 5.4.

Car type	**Landmarks**	**Route type**	**Fuel consumption (gallons)**				**GreenGPS prediction**	**Savings %**
Honda Accord 2001	H1 to Mall	Shortest	0.19	0.16	0.19	0.16	Shortest	31.4
		Fastest	0.22	0.23	0.25	0.22		
	H1 to Gym	Shortest	0.19	0.20	0.19	0.18	Shortest	19.7
		Fastest	0.21	0.23	0.22	0.25		
Ford Taurus 2001	H2 to Rest.	Shortest	0.24	0.23	0.23	0.22	Shortest	26
		Fastest	0.3	0.28	0.29	0.29		
Toyota Celica 2001	H2 to Work	Shortest	0.18	0.16	0.18	0.17	Fastest	10.1
		Fastest	0.17	0.14	0.16	0.15		
Nissan Sentra 2009	H3 to CUPHD	Shortest	0.14	0.15	0.15	0.15	Fastest	8.4
		Fastest	0.13	0.13	0.14	0.14		
Honda Civic 2002	H4 to Work	Shortest	0.33	0.32	0.33	0.3	Fastest	18.7
		Fastest	0.25	0.28	0.27	0.24		

Table 5.4: Table showing fuel consumptions for the various roundtrips between different landmarks

We observe from Table 5.4 that the fuel-optimal route for destinations of the Honda Accord and Ford Taurus was the shortest route, whereas, for the other three destinations it was the fastest route. Hence, picking the shortest or fastest routes consistently is not optimal. To confirm that the differences in fuel consumption between the compared routes are not due to measurement noise, we tested the statistical significance of the difference in means using the two paired *t-test*. The test yielded that the differences are statistically significant with a confidence level of at least 90%. The average savings (by choosing the correct route over the alternative) for each pair of landmarks and car are summarized in Table 5.4.

Comparing the total fuel consumed on the optimal route to the average of that consumed on the shortest route and fastest route (assuming the driver guesses at random in the absence of GreenGPS), the savings achieved are roughly 6% over the shortest path and 13% over the fastest, which is consistent with data we reported earlier in the feasibility study.[3]. This is by no means statistically significant, since only a handful

[3]The feasibility study used different routes from those reported above

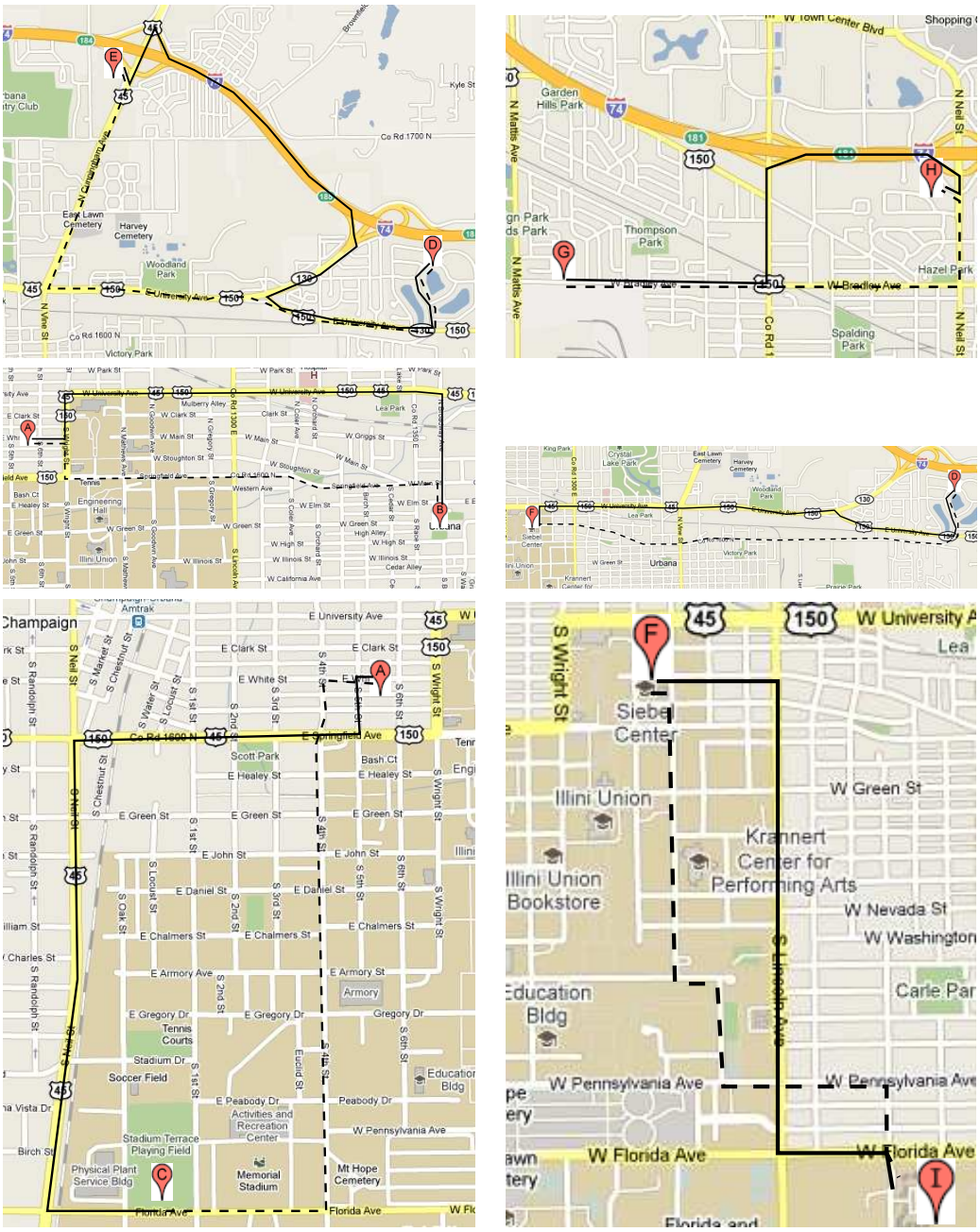


Figure 5.14: Figure showing the landmarks and corresponding shortest and fastest routes

of routes were used in the experiments above, but it nevertheless shows promise as a proof of concept.

5.7 Lessons Learned

This section presents, in its two respective subsections, a brief discussion of our experiences with the GreenGPS service and the limitations of the current study.

5.7.1 Experiences with GreenGPS

Several lessons were learned from GreenGPS, as an example of participatory sensing applications. First, we observed that data cleaning is an important problem and is application dependent. We had several occasions when several fields were missing from the data. For example, the GPS sometimes failed to communicate with the DashDyno and the location fields were then empty. A simple scheme (integrated with the DashDyno specific data formatter module of PoolView) was used to filter complete datasets from those that were missing values. Another data-related issue was the presence of noise in the data. For example, in our setup, we observed that (in some car models) whenever the GPS communicated with the DashDyno, the *fuel rate* measurement had a large spike. This was likely due to improper use of sensor IDs, which led to data overwriting. Solutions have to be developed that filter the noise at the source. For example, we developed a simple filter (integrated with the data formatter module of PoolView) that removes outliers from the data before storing it. An application-specific challenge was observed due to the slight variations in the OBD-II standards among different cars. For example, we observed that the Toyota Prius (by default) outputs the speed and fuel measurements in the metric system, rather than the Imperial system (which happens to be the default for the remaining cars in our dataset). It is harder to propose generic solutions to such problems. They suggest, however, that unlike small embedded systems, participatory sensing applications involve a much larger number of heterogeneous components (e.g., different car types in GreenGPS). As such components interact with each other or with aggregation services, subtle compatibility problems will play an increasing role. Troubleshooting techniques are needed that are good at identifying problems resulting from unexpected or bad interactions among different individually well-behaved components. This is to be contrasted, for example, with debugging tools that attempt to find bugs in individual components.

Finally, another lesson learned relates to the initial experimental deployment of participatory sensing systems. A major hurdle in getting participatory sensing systems off the ground is to provide the right incentives to the individuals (who are part of the system) [93]. We believe that the initial deployment, which

tends to be sparse, should be carefully designed in order to provide incentives for larger adoption. It should therefore be useful from the very early stages.

5.7.2 Limitations of Current Study

Apart from the limitations arising from the small size of the data set, discussed earlier, we also make the following observations. As expected, the main factors affecting fuel consumption of a vehicle on a path are the average speed, the speed variability (estimated by averaging the speed squared), and the engine idle time (estimated from the number of stop signs and stop lights on the path). A limitation of the study is that we did not explore the use of real-time traffic conditions for purposes of fuel estimation. Rather, we opted to use statistical averages of speed, speed variability and idle time. It is easy to see how such statistical averages can be computed for different hours of the day and different days of the week given a sufficient amount of historical data, yielding expected fuel consumption (in the statistical sense of expectation). The outcome is that individual trips may differ significantly from the statistical expectation. However, by consistently following routes that have a lower expected fuel consumption, savings will accumulate in the long term. Drivers may think of GreenGPS as a long-term investment. Short-term results may vary, but long-term expectations should tend to come true.

A limitation of the study, as discussed in Section 5.3, is that the selection of cars used in our current study (mostly compact and mid-sized sedans) result in a generalization hierarchy that ignores the car *class* (currently incorporates only car make and year). Future deployments will consider a broader range of vehicles, such as SUVs, minivans, and light trucks. The data from these deployments will be used to recompute a better generalization hierarchy.

In order to achieve the next level of optimization, a next generation of GreenGPS can take into account the real-time situation. Our experience reveals, not surprisingly, that the degree of congestion plays the largest role in accounting for fuel consumption variations among individual trips of the same vehicle. On lightly-utilized streets, another main factor is the degree to which traffic lights are synchronized. Lack of synchronization accounted for up to a 50% increase in fuel consumption in our measurements.

Another limitation of the current service is that it does not properly account for turns. Turns on the path add fuel consumption, often because delays in the turn lane differ from those in the through lane. In

particular, our measurements show that left turns may add a considerable amount of delay to a path. Hence, routing should account for the type of turn as well.

Finally, we expect that fuel savings in larger cities will be higher than those reported in our evaluation, both due to the larger variability in traffic conditions that could be taken advantage of, and because of the larger connectivity which offers more alternatives in the choice of route. With the above caveats, we believe that the study remains of interest in that it explores problems typical to many participatory sensing applications, such as overcoming conditions of sparse deployment, adjusting to heterogeneity, and living with large day-to-day errors towards estimating cumulative properties. The GreenGPS study could therefore serve as an example what to expect in building similar services, as well as a recipe for some of the solutions.

5.8 Conclusions

In this chapter, we developed a solution approach for initial participatory sensing application deployments, when the data collected are sparse and high-dimensional and modeling the complex phenomenon poses a challenge. We illustrated a solution approach using a novel navigation service, called GreenGPS, that computes fuel efficient routes. This service relies on OBD-II data collected and shared by a set of users via PoolView. Lessons were described that extrapolate from experiences with this service to broad issues with participatory sensing service design in general. We also show that significant fuel savings can be achieved by using GreenGPS, which not only reduces the cost of fuel, but also has a positive impact on the environment by reducing CO_2 emissions. An important issue addressed was surviving conditions of sparse deployment. GreenGPS achieves this by using a hierarchy of models developed in this Chapter to estimate the fuel consumption, and shooting for models that are unbiased, if not accurate. Our future work will address the challenges associated with real-time prediction, as well as experiences from a longer-term deployment.

Chapter 6

Related Work

We divide the related work section into three parts, the first part will summarize the literature in human activity identification. In the second part, we will look at various participatory sensing applications and finally review privacy related work.

6.1 Human Physical Activity Identification

We will first describe the related work for identification of basic human activities, such as walking, running, and writing. Then, we will describe the related work corresponding to identification of activities of daily living (ADLs).

6.1.1 Basic Human Activity Identification

There has been considerable work on activity identification using wearable computers. We discuss a representative few in this section. We present a few accelerometer based wearable computers that identify human activities. We will then examine wearable computers that use other sensors.

A wearable jacket and a sensor badge have been developed in [41] which sense perambulatory activities for context awareness in a human. The jacket uses knitting techniques to form stretch sensors positioned to measure upper limb and body movement. The sensor badge uses 2-axis accelerometers to identify different postures of the human - *standing, sitting, lying, walking* and *running*. The data from the accelerometer is sampled at 20 samples/second to differentiate between the above postures. The sensor jacket is used to detect the posture and movements of the user by using *knitted stretch sensors* and *knitted conductive tracking.* Feature vector based activity identification using accelerometers to identify activities such as *standing, walking* and *running* has been presented in [92]. A *cyberjacket* using context sensors incorporating

a tourist guide application has been built in [91]. A crossbow [30] ADXL202 2-axis accelerometer is used to analyze the user's behavior and classify them into different states, such as *sitting* and *walking*. The application program could register an interest in the event that the users activity changes from walking to sitting. This could trigger the main processor and display relevant information to the user. A system with accelerometers on pants attached to the laptop to interpret the raw sensor data using Kohonen maps and machine learning techniques for learning the user's activities is presented in [69]. The system consists of ADXL05 Analog devices two-axis accelerometers connected to a PIC microprocessor, which sends data to a laptop via. the serial port. They were able to distinguish between various classes of movements, such as *walking, sitting, running, jumping, climbing stairs, descending stairs* and *riding a bicycle.* A wrist worn fall detector for elderly individuals has been developed in [32]. The wrist device uses a three-axial accelerometer to measure the acceleration and based on the norm of the acceleration measurements, the fall is detected. A method to translate gesticulation into musical performance to express the performer's emotion is presented in [96]. The system consists of three dimensional acceleration sensors, a MIDI sound source and a computer. This system controls the musical system directly by human gesture. Techniques for processing data from accelerometers which enable the wearable computer to determine user's activity are presented in [90]. They use a clustering algorithm - a neural network to infer what the user is doing. Magnetic field, angular rate and gravity sensors (MARG sensors) are used in [12] and [13] to determine the posture of an articulated body. In the system, orientation relative to an earth-fixed reference frame of each limb segment is individually determined through the use of an attached MARG sensor. Orientations are used to set posture of an articulated body model. An inertial gesture recognition framework composed of 3 parts is presented in [17]. The first is a set of six-axis wireless inertial measurement unit to fully capture the three-dimensional motion. The second is a gesture recognition algorithm for analyzing data and categorizing them axis-by-axis as simple motions with magnitude and duration. The third allows an application designer to combine recognized gestures both concurrently and consecutively to create specific composite gestures that can be set to trigger output routines. A real time motion tracking system using sensors built from tri-axis microelectromechanical accelerometers, rate gyros, and magnetometers is presented in [119]. A Kalman filter based fusion algorithm was applied to obtain dynamic orientations and further positions of segments of the subject's body. In [14], a low-cost/low-power wearable motion tracking system is developed, based

on integrated accelerometers, called MOCA (motion capture with accelerometers). The system is composed of sensing units connected to a control/acquisition board, responsible for reading and preprocessing data, and a mobile terminal running the recognition algorithm.

Our work on the smart jacket introduces the next generation of wearable computing systems in which wired networks and centralized processing are replaced with wireless sensors individually equipped with their own microprocessors, memory, and radio devices. This decentralization offers more flexibility, scalability, and independence within the computing platform. It also allows for disconnected operation of the smart attire, where the user need not be in a specialized environment. The wireless nature of our system and its added flexibility allow subsystems of sensors from articles of clothing (such as pants, shirts, and shoes) to communicate and form a single sensor group with the purpose of providing additional functionality. Further, the algorithm that we develop outperforms the existing approaches (which are feature vector based).

6.1.2 Identification of Activities of Daily Living

In the previous section (Section 6.1.1), we have introduced several papers that have developed techniques for identifying basic activities, such as walking, running, sitting, and lying [41, 92, 91, 96, 90]. In this section, we will focus on related work that is concerned with identification of a broader set of high level activities, such as cooking, driving, and eating. As we have seen earlier, these activities are termed *activities of daily living*.

A Gaussian Mixture Model (GMM) based approach combined with a finite state machine was developed in [63] for the purpose of identification of early morning bathroom activities, such as washing face, brushing teeth, and shaving. The sensor data input used for the identification of the above activities was an accelerometer strapped to the wrist. In their previous work [62], the authors describe an integrated system that combines sensors embedded in home as well as wearable sensors for the collection of data for activity identification and labeling of the collected data. We identify a broader set of activities and use an existing device, i.e. the cell phone for identifying these activities.

In [87], RFIDs were used to identify ADLs. A system called Proact was built that uses inputs from RFIDs attached to various objects and a reader attached to a glove. These inputs were used to create models using *dynamic Bayesian networks*, that are then used for the identification of the activities. An approach

that augments the use of RFIDs with an accelerometer mounted on the glove with the RFID reader was presented in [99]. In the paper, RFID tag readings were used to narrow down the set of activities based on the type of object being used. Then, features were extracted from the accelerometer data, after which three different approaches for classification were used, namely Naive Bayes, Hidden Markov Models, and Joint Boosting. In contrast to the above paper, our work uses existing sensors from cell phones and avoids the use of cumbersome devices like a glove with an RFID reader attached and does not take input from the sensors embedded in the environment.

An approach to identify a minimal set of sensors for the purpose of identification of eating and meal preparation was presented in [71]. In the paper, the adaptive boosting (AdaBoost) classifier was used for separating eating and associated tasks from other activities. Our work addresses the identification of a larger set of activities with a limited set of sensors that are available on the cell phone.

State change sensors installed on various household objects such as doors, drawers, and refrigerators are used to collect sensor data for the purpose of the identification of activities of daily living, such as cooking, shopping, washing, and bathing in [117]. A self-adaptive neural network called Growing Self-Organizing Maps is used for the purpose of activity identification. In contrast to our work, the above paper used inputs from objects tagged with state change sensors.

A specialized device that records various sensor readings, such as the microphone, light, accelerometer, and barometer was developed in [27]. An on-device inference algorithm that identifies activities such as walking, sitting, climbing stairs, and brushing teeth was also presented. Our work on the other hand identifies a broader set of activities and does not require a specialized device to be used.

6.2 Participatory Sensing

In this section, we will discuss various participatory sensing applications followed by a few architectures that were proposed for participatory sensing and fuel efficiency related applications.

6.2.1 Participatory Sensing Applications

Participatory sensing applications have recently been described as an important emerging category of future sensing systems [2]. Several applications have been developed and deployed, some examples include a

participatory sensor network to search and rescue hikers in mountains [58], vehicular sensor networks (cars with sensor nodes) such as CarTel [61], that deployed sensor nodes in cars, and sensor networks embedded in individuals attire [47], cyclist networks (BikeNet) [35], cellphone camera networks for sharing diet related images (ImageScape) [94], cellphone networks for media sharing (MMM2) [31], sensor networks for parking space monitoring [74], and image search using mobile phones [113]. In this thesis, we focus on the general data analysis tools required for the development of participatory sensing applications, as opposed to developing a specific application.

6.2.2 Participatory Sensing Architectures

An architecture for participatory sensing, called *Partisans*, has been proposed in [86]. In that paper, the main challenges addressed are those of data verifiability and privacy. In contrast to our work, the approach assumes a trusted third party. A similar trust model was assumed in [68]. Recently, an architecture and a set of tools for data collection and analysis from weather centers are being developed by CASA [29]. These architectures do not consider privacy of the sensor data being shared (or assume the presence of a trusted third party). Further, they assume that data density is high (although, we have shown that initial participatory sensing deployments will be sparse and the data density will be low).

6.2.3 Fuel Efficiency Related Applications

A comprehensive study that provides optimal route choices for lowest fuel consumption is presented in [37]. In the paper, fuel consumption measurements are made through the extensive deployment of sensing devices (different from the OBD-II) in experimental cars. These fuel consumption measurements are then used to compute the lowest fuel consumption route. In contrast to the work in [37], we use a sparse deployment to build mathematical models for predicting fuel consumption for other streets and cars. In [21], the influence of driving patterns of a community on the exhaust emissions and fuel consumption were studied. Feedback was provided to the community regarding the driving patterns to cut back on the fuel consumption and exhaust. A driver support tool, FEST, was developed in [33]. FEST uses sensors installed in the car along with a software to determine the driving behavior of the driver and provide real-time feedback to the individual for the purpose of reduction in fuel consumption. An extension to FEST that includes more experiments

and further evaluation can be found in [104]. A feedback control algorithm was developed in [97] that determines speed of automobiles on highways with varying terrain to achieve minimal fuel consumption. An extension to the work in [97] was developed in [57]. In [57], suggestions of driving style to minimize fuel consumption were made for varying road and trip types (e.g. constant grade road, hilly road). The problem was formulated using a control theoretic approach.

UbiGreen [44] is a mobile tool that tracks an individual's personal transportation and provides feedback regarding their CO_2 emissions.

In a separate study [64], it was shown that rising obesity has a significant impact on the total fuel consumption in the US. Models were developed that studied the impact of obesity on the amount of fuel consumed in passenger vehicles.

Our work in Chapter 5 represents the first participatory sensing service that aims at improving fuel consumption. Using data collected from volunteer participants, models are built and continuously updated that enable navigation on the minimum-fuel route.

6.3 Privacy

Privacy is an important problem in Internet based applications, as pointed out in [51]. Several privacy approaches and algorithms have been developed, which span various fields of computer science. In this section, we will provide a comprehensive summary of related work with respect to sensor data (it is outside the scope of this thesis to survey all aspects of privacy). We look at privacy techniques in the data mining literature (also called as privacy preserving data mining), as it is most closely related to our work.

We classify past work into four broad categories: (i) Data anonymization, (ii) Random perturbation (iii) Randomized response, and (iv) Secure multi-party computation. These techniques presented below can be leveraged in future incarnations of our architecture.

6.3.1 Data Anonymization

The concept of data anonymization is one where individuals share the data without revealing their true identity. A typical approach is one of *k-anonymity*, where the model is that the data released by an individual cannot be distinguished from at least $k-1$ other individuals. This k-anonymity model was introduced

in [100] and it was shown that it can protect against certain types of attacks (e.g. unsorted matching attack, complementary release attack, and temporal attack). An optimal and practical k-anonymity approach is presented in [15], which extends existing approaches such as [110, 95]. The work in [15] presents a k-anonymity approach that hides minimal data from a given dataset (for an individual) using a tree-based pruning technique. Privacy preservation using k-anonymity techniques for pattern mining is proposed in [8], which was extended to support mining frequent itemsets in a privacy preserving manner in [7]. A real-time k-anonymity based approach for social network data sharing is presented in [16]. Recently, k-anonymity techniques have been extended to address location privacy (in particular) [19, 18, 43, 118]. The concept of *mix zones* with trusted middleware was used in [19, 18, 43] to achieve location privacy. A distributed k-anonymity based protocol for location privacy was proposed in [118].

The k-anonymity approach has a drawback, in that, it may still reveal certain private information regarding the individual (sharing data). For example, consider anonymized GPS data. Anonymized GPS data may still reveal the identity of the user (e.g. Identifying the home location and work location can narrow down the identity of the individual and in many cases reveal it too). Approaches such as [19, 18, 118] rely on trusted middleware, which in our approach is absent.

6.3.2 Randomized Perturbation Based Techniques

The general idea behind these techniques is to perturb the individual data being shared in such a manner so that single data items "appear" to be random values to the data miner and an external entity cannot draw inferences about individual private data with a certain degree of confidence. One of the first such approaches was proposed in [5]. In this paper, each client has a numerical data item x_i, and an external server wants to compute the distribution of the data items over all clients. The clients randomize their data items by adding a random number r_i drawn independently from a known distribution such as a uniform or Gaussian distribution, where the mean of the distribution is known. The server collects the values $x_i + r_i$ and reconstructs the distribution of x_i's using the Bayes' rule to estimate a posterior distribution function. Further, the authors' of [5] provide a method to quantify privacy. This method is based on how closely the original values of a modified attribute can be estimated. If the original value x can be estimated with a $c\%$ confidence that it lies in the interval $[x_1, x_2]$, then the interval width $(x_2 - x_1)$ is defined as the amount of

privacy at $c\%$ confidence interval.

The work in [5] was extended by the authors' of [4]. In [4], a reconstruction algorithm was proposed that converges to the maximum likelihood estimate of the original distribution. The reconstruction algorithm is based on the Expectation Maximization (EM) algorithm. Further, metrics are developed that quantify privacy and information loss based on *differential entropy*.

Several papers, [39, 66, 58], extended the technique presented in [5]. These papers show that privacy breaches occur under certain conditions, when the randomized perturbation approach of [5] is used. They then develop solutions to prevent such breaches. In [39], it has been shown that the technique of [5] is vulnerable when a set of items are shared (instead of a single value). A *randomized operator* was developed that prevents privacy breaches when itemsets are shared. The privacy preserving properties of [5] were studied in [66] and it was shown that, in certain cases, the data perturbation technique of [5] fails to preserve privacy. The paper uses the properties of random matrices and spectral filtering techniques to retrieve the original data from the distorted data set. Although, the paper fails to mention the "conditions" under which such breaches happen. Further studies in [58] revealed that a key factor that leads to privacy breaches of the technique in [5] is based on data correlations. They propose a Principal Component Analysis (PCA) based technique to estimate the original data given the distorted data. Then, they propose a scheme that perturbs the data with random noises which are "similar" to the original data. This scheme is proven to be privacy preserving when their technique of PCA is used to reconstruct the original data.

Our privacy preservation scheme differs from the above approaches in that we consider time-series sensor data, which are correlated (in the time dimension). The correlations in time-dimension can be exploited to infer the trends or estimate the original sensor data stream (using techniques like PCA). Hence, using techniques that do not take into account the correlation of sensor data in the time dimension will not work.

On the lines of the above additive random data perturbation approach, multiplicative data perturbation approaches have been developed, [67, 25]. In [67], two multiplicative data perturbation schemes were proposed to do privacy preserving data mining. One approach is to generate random numbers which have a mean of one and a small variance, and then multiplying the original data by this noise. The second approach is a bit more complicated. This technique uses a logarithmic transformation on the data, combined with the computation of a covariance matrix of the transformed data. A random number with a covariance

similar to that of the transformed data is generated. The transformed data item and the random noise are added and an antilog of the result is shared. These techniques were used in masking the income data from the 1990 Internal Revenue Service (IRS) 1040 Income Tax Return file! A geometric rotation based approach for *data classification* is presented in [25], that preserves certain geometric properties of datasets when transforming them. This transformed dataset is shared with external entities. As certain properties of the dataset are preserved, it is possible to obtain classifiers from the perturbed datasets. Three different geometric transformation techniques are presented for dataset transformation.

The above multiplicative based data perturbation approaches also do not consider the correlations in the time domain. Our approach utilizes additive methods, but it may be possible to use a multiplicative approach (that considers the correlations within the sensor data stream).

6.3.3 Randomized Response Based Techniques

The randomized response technique was first introduced by Warner [109] as early as 1965. It was introduced to solve a survey problem, which can be stated as follows: to find an estimate of the percentage of people in a given population that has a sensitive attribute X. Two models were proposed in [109] that estimate the above percentage without revealing the answer to whether an individual has the sensitive attribute. One model asks each respondent two related questions[1], which are as follows: (i) Do you have the sensitive attribute X? (ii) Do you not have the sensitive attribute X? Note that, the answers to the questions are opposites of each other. A randomizing device is designed that chooses the first question with probability θ and the second with $1 - \theta$. The external entity only learns the answer to the question, i.e. a "yes" or a "no" and not the question that was answered. The estimate of the percentage of people who have the attribute X is obtained by solving the following equations:

$$P^*(X = yes) = P(X = yes) \times \theta + P(X = no) \times (1 - \theta)$$

$$P^*(X = no) = P(X = no) \times \theta + P(X = yes) \times (1 - \theta)$$

In the above equations, $P^*(X = yes)$ is the proportion of the "yes" responses from the survey, and

[1]We refer you to the paper for the discussion on the second model

$P(X = yes)$ is the estimated proportion of the "yes" responses to the actual sensitive questions. This idea was extended in [38] to preserve privacy while mining categorical data (instead of numerical data). The main idea in the paper is to use a class of *randomization operators*. The randomized response scheme of Warner [109] was extended to a multiple-attribute data set in [34]. This means that instead of a single sensitive attribute X, there are multiple sensitive attributes, $X_1, X_2, \cdots X_n$. The solution approach is a straight forward extension to the randomized response scheme presented in [109].

Such randomized response approaches are not useful in computing community statistics from arbitrary time-series sensor data. Our goal is to accurately compute community statistics from arbitrary time-series sensor data while preserving the privacy of an individual.

6.3.4 Secure Multi-party Computation

The general idea behind this class of techniques is to use cryptographic methods to achieve privacy. A comprehensive treatise on the basic results of secure multi-party computation are presented in [52]. Several protocols that work under reasonable assumptions and their corresponding proofs are presented in this work. A general secure two-party function evaluation technique was developed in [115]. This technique is based on expressing the function $f(x, y)$ as a circuit and encrypting the gates for secure evaluation. These secure computation techniques have been used in [65] for developing methods for privacy preserving distributed mining of *association rules*. Association rules reflect frequent data items that are *associated* with each other, [56]. By this we mean that the data items occur together frequently. A cryptographic approach that uses the property of exponentials was proposed in [114] for the classification of customer data. The proposed approach achieves the classification without any loss of accuracy. The main idea of this approach is to use the mathematical properties of exponentiation in a cryptographic setting, that allows a data miner to combine encrypted results received from customers to calculate the desired result.

The problem with this approach is its significant overhead that requires a large number of pairwise exchanges between users in the community. Such exchanges do not scale when users are not available simultaneously for purposes of completing the computation, or when the number of users involved changes dynamically.

Chapter 7

Conclusions

In this chapter, we will first summarize the conclusions of this thesis, then describe the lessons learned, the impact of this work, and the directions for future work.

7.1 Conclusions

In conclusion, this thesis is a step towards the future embedded Internet. We designed and developed a novel architecture and data analysis toolset for human centric sensor networks and illustrated it with five different applications, *smart jacket*, *smart phone*, *traffic analyzer*, *weight watchers*, and *GreenGPS*. Our architecture builds on existing standard Internet tools that allow for easy deployment of various applications. It allows for individuals to easily collect sensor data from their everyday devices (e.g. smart phones, in-car GPS devices), analyze these data to identify various day-to-day activities and share these collected data in a larger community without breaching privacy (of the individual sharing data). These shared data can then be used to compute phenomena of common interest (e.g. air quality of cities).

In the course of the development of the PoolView architecture, we addressed various generic research challenges. These include (i) human activity identification, (ii) privacy preservation (while being able to compute accurate community statistics), and (iii) community data modeling. Human activity identification is a centerpiece for several personal application domains such as healthcare, social networking, entertainment, and personal record keeping. We show that existing algorithms are inadequate for the identification of activities using everyday sensing devices (e.g. smartphones). We developed a novel activity identification framework that can recognize activities of varying complexity. This framework combines a feature extraction library with HMMs (a Bayesian learning framework) to achieve activity identification. It can identify simple activities (such as walking, writing, typing) when only accelerometer sensor data are used and com-

plex activities (such as eating, cooking) when microphone sensor data are fused with the accelerometer sensor data. We demonstrated this framework using two prototypes, the smart jacket and the smartphone which utilize sensing devices that are available commercially.

The remaining two research challenges arise from the community sensing aspect of our architecture, where individuals share the data collected within a larger community (towards a common purpose). We observe that an important research challenge when sharing sensor data is to be able to preserve the privacy of the individual (sharing data). Further, addressing these privacy concerns in the absence of a trust hierarchy will enable the grassroots deployment of community sensing applications. We developed a novel privacy preserving technique (using data perturbation) that allows for individuals to share time series sensor data within a community such that their privacy is preserved against common reconstruction attacks, whereas it is possible to accurately reconstruct the community statistics. We applied this algorithm in the context of two applications, traffic analyzer that computes traffic related statistics from perturbed GPS sensor data shared by individuals and weight watchers which computes weight related statistics from perturbed weight data shared by individuals

Finally, we show that in a lot of community sensing applications, a common problem is the lack of sufficient sensor data to map phenomena of interest. We proposed a novel method to model phenomena of interest from relatively sparse sparse measurements of high-dimensional spaces. We illustrated this method using a novel navigation application, GreenGPS, that computes fuel efficient routes for vehicles.

7.2 Lessons Learned

In this section, we will first describe a few lessons learned from this thesis and then the limitations of the current architecture and the corresponding tools. Over the course of the development of PoolView's architecture and the corresponding applications, we observed that the amount of raw sensor data required to bridge the gap between human decision needs and the data increases exponentially with the complexity of the needs. For example, the identification of activities of a single individual (using the smart jacket) requires much lesser data than computing a fuel efficient route for a given car (using GreenGPS). This necessitates the development of a data collection framework that is automated and decentralized (allowing data collection from multiple devices across a large population).

Another important lesson that we learned from our deployments is that of incentives during initial deployments of participatory sensing applications. As typical participatory sensing applications require a large amount of data to realize the utility of the application, the right incentives have to be provided to individuals (who are part of the system).

Another lesson learned relates to the heterogeneity of the devices that are used for data collection. We observed from our deployments that different types of heterogeneity exist. First, the same sensor can have different characteristics. For example, an accelerometer can be manufactured by different companies and may have different characteristics, such as sensitivity, accuracy, precision, and errors. Second, the same sensor can behave differently depending on the interactions with other components in a system. For example, even though the accelerometers in MicaZ motes and the Nokia N95 are very similar in their characteristics, it is not possible to sample the accelerometer on Nokia N95 at a constant rate. Finally, sensing devices can interact differently in different environments. For example, a Honda Accord outputs OBD sensor measurements in the Imperial system, whereas the Toyota Prius outputs it in the metric system.

There are several limitations to the work done in this thesis. First, PoolView architecture assumes *data verifiability*. Data verification is the process where data are checked for accuracy and inconsistencies. As sensor data are generated in an automated manner, data verification needs to be an integral part of PoolView. In the current version of PoolView, simple data filtering techniques are integrated to denoise the raw sensor data. Further, PoolView does not address energy related issues. Energy (for sensing and collection of data) will be of primary importance, when the bottleneck to sense a piece of data is the energy consumed per sensed bit (as opposed to the computation power or the precision of the sensor).

A limitation of the current activity identification framework is that when multiple activities are performed together, it fails to differentiate the activities. For example, eating and watching TV were confused by our activity identification framework because they are typically performed together. This limitation can be addressed in two ways, one is to extend the HMM approach by adding further information in the model. For example, we do not consider the location information in the model, which can be used to narrow the types of activities being performed at a given location (e.g. A graduate student is more likely to cook at home than in her workplace). Second, we believe that HMMs have fundamental limitations in modeling an activity and that a completely different technique is required to achieve activity identification. For example,

data mining techniques such as the Apriori or the Prefix span algorithms [56] may be useful to mine the raw sensor data to identify human activities.

In Chapter 3, we observed that a lot of data is required to identify activities accurately. A question that needs to be addressed is if it would be possible to develop a generic model for an activity (using a small amount of data) and then evolve these models in such a way that they adapt to an individual. An approach to this problem is through the use of *semi-supervised* learning [24].

A generic limitation of current activity identification frameworks (including the one we developed) is that they need to be adapted to a specific user. The learning frameworks (e.g. Bayesian) rely on "training" data to build models, which are then used to identify an activity. Activities across different users may be significantly different and in some cases, they are different for the same individual (over time). For example, an individual who hurt her leg will have different gait from when she was normal. The broader question then is - "How do we develop an activity identification framework that does not utilize a learning based algorithm?"

Another limitation is with regard to the perturbation technique used for privacy preservation. Our algorithm protects against common reconstruction techniques such as PCA and spectral filtering. There are no fundamental guarantees provided on the privacy of the data being shared.

A space that we have not yet explored is when the noise models used for data perturbation are dynamic (continuously evolving). For example, consider a social network graph, where a single model does not suffice to describe the graph. In such a case, how does an individual perturb the data and how are community statistics reconstructed?

Finally, the modeling technique that we proposed in this thesis will not work well when data are perturbed. The error in modeling will be significantly higher when the data are perturbed. This means that we need a different privacy preservation algorithm which does not rely on data perturbation.

Yet another limitation of the modeling technique is when individuals (sharing data) are lying about the data being shared. Techniques such as comparing with spatio-temporal correlated community sensor data [86] will not work in this case. Since, the data are sparse, it becomes hard to validate the data. Further, our modeling technique assumes linear regression models (to compute the model coefficients) and simple categorical parameters to split the data into sub-cubes (e.g. car make, year, class).

7.3 Impact

The work in this thesis has been published at various top systems conferences such as MobiSys and SenSys. The human activity identification frameworks were published and presented at MobiSys [47] and BSN [46]. The privacy work presented in Chapter 4 was published and presented at SenSys [49]. Finally, GreenGPS was published and presented at MobiSys [48].

I have been *invited* to give talks on smart attire to various health care related workshops (WSNHC 2007, WAST 2008) and industrial labs (Motorola labs). At WAST 2008, a workshop organized for bringing together caregivers and computer science researchers, my work has been described as an *awesome* technology designed to improve the lives of older adults. My work featured in blog articles on Crossbow, was used as teaching material in courses at various universities such as Dartmouth (CS88/188), Duke (ECE256), and Washington University St. Louis (CS537s). In collaboration with Motorola, I developed a novel health care monitoring framework which was showcased at the Continua Health Alliance (an industry consortium of smart healthcare technologies) summit in 2008. The smartphone work was done as part of my internship at Robert Bosch research center in Pittsburgh and has been published as an internal report. A patent in this regard has been filed at the United States patent office.

For the work on GreenGPS, I have been awarded the **Siebel Scholar Fellowship** for 2009-2010, which is awarded annually for academic excellence and demonstrated leadership to the top 80 students from the world's leading graduate schools. It featured as the main news article on the homepages of the CS department, College of Engineering, and the Illinois Informatics Institute at UIUC. The work on traffic analyzer has been integrated with Microsoft research's SensorMap. The privacy work is being used as course material at various universities, such as Portland State (CS410) and UNM (CS591).

7.4 Future Work

There are three major directions of future work which arise out of this thesis, the first pertains to personal sensing in the healthcare domain, the second is to extend GreenGPS, and the third relates to building infrastructure for the pervasive availability of community sensing.

Sensing meets Healthcare: The success of smart attire, especially in the healthcare related workshop naturally leads to the first part of future work, which is to extend and generalize the personal monitoring services developed in the beginning of this thesis. In this regard, we are currently collaborating with the College of Applied Health Sciences and the Coordinated Science Laboratory at the University of Illinois, Urbana-Champaign to integrate the personal monitoring services with real healthcare applications. Our initial aim is to investigate the correlation between the blood glucose levels of Diabetes patients with their physical activity levels. Modeling the relationship between the glucose and physical activity levels will enable future medical devices to provide insulin injection reminders to the individual at appropriate times. The broader challenges from a Computer Science perspective are to develop a set of middleware services that enable multiple sensing devices to interact with each other, discover and manage resources efficiently, and collectively monitor the health of an individual. We envision that such personal monitoring services and the corresponding middleware infrastructure developed in collaboration with medical researchers will have a wide ranging impact on the current healthcare industry and will usher in the era of a sensor networked platform for healthcare.

GreenGPS: We believe that GreenGPS holds a promising direction for future research as a participatory sensing application. We have shown that individuals can gain an average of 10% savings in fuel consumption by using GreenGPS (through the means of a preliminary deployment). As discussed in Section 5.7 of Chapter 5, there are several lessons and limitations that we learned from GreenGPS and its deployment. We wish to explore these as part of our future work and envision that GreenGPS will bring about a revolution in navigation.

Sensor Enabled Internet: Data generated from sensing devices is usually automated and can result in an exponential explosion of content on the Web, which begs the question of a clever way of organizing data. Data mining solutions will play a major role in this regard. Consider the GreenGPS application, where data collected by individuals is used to create models for predicting fuel efficiency on different streets. A generalization of this problem is one where a generic model is to be constructed given a sparse dataset and different parameters. Another interesting problem is that of search, unlike the current search engines which rely on indexing the Web pages and searching for keywords to obtain relevant information, the low-level nature of sensor data will require the distillation of information, possibly involving multiple data sources

spanning wide geographical regions. Our vision is to build upon the current research (in this thesis) and lead the next generation infrastructure for a sensor rich Web.

References

[1] AAA. National average gas prices. http://www.fuelgaugereport.com/, April 2010.

[2] T. Abdelzaher et al. Mobiscopes for human spaces. *IEEE Pervasive Computing*, 6(2):20–29, 2007.

[3] Actron. Elite autoscanner. http://www.actron.com/product_category.php?id=249.

[4] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proc. of ACM Principles of Database Systems*, pages 247–255, 2001.

[5] R. Agrawal and R. Srikant. Privacy preserving data mining. In *Proc. of ACM Conf. on Management of Data*, pages 439–450, May 2000.

[6] S. S. Alpert. A two-reservoir energy model of the human body. *The American Journal of Clinical Nutrition*, 32(8):1710–1718, 1979.

[7] M. Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi. Blocking anonymity threats raised by frequent itemset mining. In *ICDM*, pages 27–30, 2005.

[8] M. Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi. k-anonymous patterns. In *PKDD*, pages 10–21, 2005.

[9] Auterra. Dashdyno. http://www.auterraweb.com/dashdynoseries.html.

[10] AutoTap. Autotap reader. http://www.autotap.com/products.asp.

[11] AutoXRay. Ez-scan. http://www.autoxray.com/product_category.php?id=338.

[12] E. R. Bachmann, R. B. McGhee, X. Yun, and M. J. Zyda. Inertial and magnetic posture tracking for inserting humans into networked virtual environments. In *Proc. of the ACM symposium on virtual reality software and technology*, pages 9–16. ACM, November 2001.

[13] E. R. Bachmann, X. Yun, and R. B. McGhee. Sourceless tracking of human posture using small inertial/magnetic sensors. In *Proceedings 2003 IEEE International symposium on computational intelligence in robotics and automation*, pages 822–829, Kobe, Japan, July 2003. IEEE.

[14] R. Barbieri, E. Farella, L. Benini, B. Ricco, and A. Acquaviva. A low-power motion capture system with integrate accelerometers (gesture recognition applications). In *Proc. of the IEEE consumer communications and networking conference*, pages 418–423. IEEE, January 2004.

[15] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymization. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, pages 217–228, 2005.

[16] A. Beach, M. Gartrell, and R. Han. Social-k: Real-time k-anonymity guarantees for social network applications. In *PerCom*, pages 600–606, 2010.

[17] A. Y. Benbasat and J. A. Paradiso. An inertial measurement framework for gesture recognition and applications. In *Revised papers from the international gesture workshop on gesture and sign languages in human-computer interaction*, pages 9–20, 2001.

[18] A. R. Beresford and F. Stajno. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55.

[19] A. R. Beresford and F. Stajno. Mix zones: User privacy in location-aware services. In *Second IEEE Conference on Pervasive Computing and Communications Workshops*, page 127, 2004.

[20] D. M. Bevly, R. Sheridan, and J. C. Gerdes. Integrating ins sensors with gps velocity measurements for continuous estimation of vehicle sideslip and tire cornering stiffness. In *Proc. of American Control Conference*, pages 25–30, 2001.

[21] K. Brundell-Freij and E. Ericsson. Influence of street characteristics, driver category and car performance on urban driving patterns. *Transportation Research, Part D*, 10(3):213–229, 2005.

[22] J. Burke et al. Participatory sensing. Workshop on World-Sensor-Web, co-located with ACM SenSys, 2006.

[23] C. Carson and H. Kevin. The dynamics of human body weight change. *PLOS Computational Biology*, 4(3):1000045, March 2008.

[24] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[25] K. Chen and L. Liu. Privacy preserving data classification with rotation perturbation. In *Proc. of IEEE International Conference on Data Mining*, pages 589–592, 2005.

[26] Y. Chen et al. Regression cubes with lossless compression and aggregation. *IEEE Transactions on Knowledge and Data Engineering*, 18(12):1585–1599, 2006.

[27] T. Choudhury et al. The mobile sensing platform: An embedded activity recognition system. *IEEE Pervasive Computing*, 7(2):32–41, April-June 2008.

[28] D. Clark. Internet meets sensors: Should we try for architecture convergence? Presented at the Networking of Sensor Systems (NOSS) Principal Investigator and Informational Meeting, October 2005.

[29] Collaborative Adaptive Sensing of the Atmosphere. http://www.casa.umass.edu/.

[30] Crossbow Technologies. http://www.xbow.com/.

[31] M. Davis et al. Mmm2: Mobile media metadata for media sharing. In *CHI Extended Abstracts on Human Factors in Computing Systems*, pages 1335–1338, 2005.

[32] T. Degen, H. Jaeckel, M. Rifer, and S. Wyss. Speedy: a fall detector in a wrist watch. In *Proc. of the IEEE International Symposium on Wearable Computers*, pages 184–187. IEEE, October 2003.

[33] V. der Voort. Fest - a new driver support tool that reduces fuel consumption and emissions. *IEE Conference Publication*, 483:90–93, 2001.

[34] W. Du and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *Proc. of ACM SIGKDD Conf.*, pages 505–510, 2003.

[35] S. B. Eisenman et al. The bikenet mobile sensing system for cyclist experience mapping. In *Proc. of SenSys*, November 2007.

[36] EPA. Emission facts: Greenhouse gas emissions from a typical passenger vehicle. http://www.epa.gov/OMS/climate/420f05004.htm.

[37] E. Ericsson, H. Larsson, and K. Brundell-Freij. Optimizing route choice for lowest fuel consumption - potential effects of a new driver support tool. *Transportation Research, Part C*, 14(6):369–383, 2006.

[38] A. Evfimievski. Randomization in privacy preserving data mining. *ACM SIGKDD Explorations Newsletter*, 4(2):43–48, December 2002.

[39] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the SIGMOD/PODS Conference*, pages 211–222, 2003.

[40] J. Farrelly and P. Wellstead. Estimation of vehicle lateral velocity. In *Proc. of IEEE Conference on Control Applications*, pages 552–557, 1996.

[41] J. Farringdon, A. J. Moore, N. Tilbury, J. Church, and P. D. Biemond. Wearable sensor badge and sensor jacket for context awareness wearable sensor badge and sensor jacket for context awareness. In *Proc. of the IEEE international symposium on wearable computers*, pages 107–113. IEEE, October 1999.

[42] G. B. Forbes. Weight loss during fasting: Implications for the obese. *The American Journal of Clinical Nutrition*, 23(9):1212–1219, September 1970.

[43] J. Freudiger, R. Shokri, and J.-P. Hubaux. On the optimal placement of mix zones. *Privacy Enhancing Technologies*, 5672:216–234, 2009.

[44] J. E. Froehlich et al. Ubigreen: Investigating a mobile tool for tracking and supporting green transportation habits. In *In Proc. of Conference on Human Factors in Computing*, pages 1043–1052, 2009.

[45] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *Proc. of ACM SENSYS*, November 2003.

[46] R. Ganti, S. Srinivasan, and A. Gacic. Multisensor fusion in smartphones for lifestyle monitoring. In *In Proc. of BSN '10*, June 2010.

[47] R. K. Ganti, P. Jayachandran, T. F. Abdelzaher, and J. A. Stankovic. Satire: a software architecture for smart attire. In *Proc. of ACM MobiSys*, pages 110–123, 2006.

[48] R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. Abdelzaher. Greengps: A participatory sensing fuel-efficient maps application. In *In Proc. of MobiSys*, pages 151–164, June 2010.

[49] R. K. Ganti, N. Pham, Y.-E. Tsai, and T. Abdelzaher. Poolview: Privacy in grassroots participatory sensing. In *Proc. of SenSys*, pages 281–294, November 2008.

[50] Garmin eTrex Legend. www8.garmin.com/products/etrexlegend.

[51] I. Goldberg, D. Wagner, and E. Brewer. Privacy-enhancing technologies for the internet. In *COMPCON '97: Proceedings of the 42nd IEEE International Computer Conference*, page 103, 1997.

[52] O. Goldreich. Secure multi-party computation (draft). Technical report, Weizmann Institute of Science, 2002.

[53] Google. Google maps. http://maps.google.com.

[54] GPS POI. Red light database. http://www.gps-poi-us.com/.

[55] J. Gray et al. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. *Data Mining and Knowledge Discovery*, 1(1):29–54, 1997.

[56] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, second edition, 2006.

[57] J. N. Hooker. Optimal driving for single-vehicle fuel economy. *Transportation Research, Part A*, 22A(3):183–201, 1988.

[58] J.-H. Huang, S. Amjad, and S. Mishra. Cenwits: a sensor-based loosely coupled search and rescue system using witnesses. In *Proc. of SenSys*, pages 180–191, 2005.

[59] Z. Huang, W. Du, and B. Chen. Deriving private information from randomized data. In *Proc. of ACM SIGMOD Conference*, pages 37–48, June 2005.

[60] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proc. of ACM SENSYS*, November 2004.

[61] B. Hull et al. Cartel: a distributed mobile sensor computing system. In *Proc. of SenSys*, pages 125–138, 2006.

[62] N. F. Ince, C.-H. Min, and A. H. Tewfik. Integration of wearable wireless sensors and non-intrusive wireless in-home monitoring system to collect and label the data from activities of daily living. In *Proceedings of EMBS Annual Conference*, pages 28–31, 2006.

[63] N. F. Ince, C.-H. Min, and A. H. Tewfik. A feature combination approach for the detection of early bathroom activities with wireless sensors. In *Proceedings of HealthNet'07*, pages 61–63, 2007.

[64] S. H. Jacobson and L. A. McLay. The economic impact of obesity on automobile fuel consumption. *Engineering Economist*, 51(4):307–323, 2006.

[65] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed data mining of association rules on horizontally parititoned data. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1026–1037, September 2004.

[66] H. Kargutpa, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proc. of the IEEE International Conference on Data Mining*, pages 99–106, 2003.

[67] J. J. Kim and W. E. Winkler. Multiplicative noise for masking continuous data. Technical Report Statistics #2003-01, Statistical Research Division, U.S. Bureau of the Census, Washington D.C., April 2003.

[68] A. Krause, E. Horvitz, A. Kansal, and F. Zhao. Toward community sensing. In *Proc. of IPSN*, 2008.

[69] K. V. Laerhoven and O. Cakmakci. What shall we teach our pants? In *Proc. of the IEEE International Symposium on Wearable Computers*, pages 77–83. IEEE, October 2000.

[70] D. Li, I. Sethi, N. Dimitrova, and T. McGee. Classification of general audio data for content-based retrieval. *Pattern Recognition Letters*, 22(5):533–544, 2001.

[71] B. Logan and J. Healey. Sensors to detect the activities of daily living. In *Proceedings of EMBS Annual Conference*, pages 5362–5365, 2006.

[72] MapQuest. http://www.mapquest.com.

[73] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *Proc. of ACM SENSYS*, Novemeber 2004.

[74] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe. Parknet: drive-by sensing of road-side parking statistics. In *MobiSys '10: Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 123–136, June 2010.

[75] R. E. Mickens, D. N. Brewley, and M. L. Russell. A model of dieting. *SIAM Review*, 40(3):667–672, September 1998.

[76] E. Miluzzo et al. Sensing meets mobile social networks: The design, implementation and evaluation of the cenceme application. In *Proc. of SenSys*, November 2008.

[77] D. Minnen, T. Starner, J. A. Ward, P. Lukowicz, and G. Troster. Recognizing and discovering human actions from on-body sensor data. In *Proc. of the IEEE International Conference on Multimedia and Expo*, July 2005.

[78] National Aeronautics and Space Administration (NASA). Landsat data. http://landsat.gsfc.nasa.gov/data/.

[79] New York State Office of the Aging - Toolkit for Caregivers. http://www.aging.ny.gov/Caregiving/Toolkit/7CaringforYourParentsFactSheetsinEnglish/ChecklistofActivitiesofDailyLiving.pdf.

[80] Nic Roets. Gosmore. http://wiki.openstreetmap.org/wiki/Gosmore.

[81] Nokia S60 SDK. http://www.forum.nokia.com/.

[82] OpenStreetMap. Openstreet map. http://wiki.openstreetmap.org/.

[83] A. V. Oppenheim, R. W. Schafer, and J. R. Buck. *Discrete-time signal processing (2nd ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1999.

[84] Owen Brotherwood. Symbtelm. http://sourceforge.net/apps/trac/symbtelm/.

[85] S. Papadimitriou, F. Li, G. Kollios, and P. S. Yu. Time series compressibility and privacy. In *In Proc. of VLDB '07*, pages 459–470, September 2007.

[86] A. Parker et al. Network system challenges in selective sharing and verification for personal, social, and urban-scale sensing applications. In *Proceedings of HotNets-V*, pages 37–42, 2006.

[87] M. Philipose, K. P. Fishkin, M. Perkowitz, D. J. Patterson, D. Fox, H. Kautz, and D. Hahnel. Inferring activities from interactions with objects. *IEEE Pervasive Computing*, 3(4):10–17, October-December 2004.

[88] Polisher Research Institute
. http://www.abramsoncenter.org/PRI/documents/IADL.pdf.

[89] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–286, February 1989.

[90] C. Randell and H. Muller. Context awareness by analysing accelerometer data. In *Fourth International Symposium on Wearable Computers*, pages 175–6, Atlanta, GA, October 2000. IEEE.

[91] C. Randell and H. L. Muller. The well mannered wearable computer. In *Proc. of Personal and Ubiquitous computing*, volume 6, pages 31–36, February 2002.

[92] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman. Activity recognition from accelerometer data. In *Proc. of the Innovative Applications Conference on Artificial Intelligence*, pages 1541–1546, July 2005.

[93] S. Reddy, D. Estrin, and M. Srivastava. Recruitment framework for participatory sensing data collections. In *To Appear in Proc. of Intnl. Conference on Pervasive Computing*, 2010.

[94] S. Reddy et al. Image browsing, processing, and clustering for participatory sensing: Lessons from a dietsense prototype. In *Proceedings of Embedded Networked Sensors, EmNets '07*, pages 13–17, 2007.

[95] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *PODS '98: Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, page 188, 1998.

[96] H. Sawada and S. Hashimoto. Gesture recognition using an acceleration sensor and its application to musical performance control. *Electronics and Communications in Japan*, 80(5):452–459, 1997.

[97] A. B. Schwarzkopf and R. B. Leipnik. Control of highway vehicles for minimum fuel consumption over varying terrain. *Transportation Research*, 11(4):279–286, 1977.

[98] S.-A. Selouani, H. Tolba, and D. O'Shaughnessy. Auditory-based acoustic distinctive features and spectral cues for robust automatic speech recognition in low-snr car environments. In *Proc. of the American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 91–93, 2003.

[99] M. Stikic, T. Huynh, K. V. Laerhoven, and B. Schiele. Adl recognition based on the combination of rfid and accelerometer sensing. In *Proceedings of Pervasive Health*, pages 258–263, 2008.

[100] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[101] Traffic. Real-time traffic conditions. http://www.traffic.com/.

[102] H. E. Tseng. Dynamic estimation of road bank angle. *Vehicle System Dynamics*, 36(4-5):307–328, 2001.

[103] US Census Bureau. Tiger database. http://www.census.gov/geo/www/tiger/.

[104] M. van der Voort, M. S. Dougherty, and M. van Maarseveen. A prototype fuel-efficiency support tool. *Transportation Research, Part C*, 9(4):279–296, 2001.

[105] P. H. Veltink, H. B. J. Bussmann, W. de Vries, W. L. J. Martens, and R. C. V. Lummel. Detection of static and dynamic activities using uniaxial accelerometers. *IEEE Transactions on Rehabilitation Engineering*, 4:375–385, December 1996.

[106] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy. Psfq: A reliable transport protocol for wireless sensor networks. In *Proc. of the ACM International Workshop on Wireless Sensor Networks and Applications*, September 2002.

[107] R. Want. You are your cell phone. *IEEE Pervasive Computing*, 7(2):2–4, April–June 2008.

[108] J. A. Ward, P. Lukowicz, G. Troster, and T. E. Starner. Activity recognition of assembly tasks using body-worn microphones and accelerometers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1553–1567, October 2006.

[109] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, March 1965.

[110] W. E. Winkler. Using simulated annealing for k-anonymity. Technical report, U. S. Census Bureau, 2002.

[111] XML REC. http://www.w3.org/tr/rec-xml/.

[112] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 13–24, 2004.

[113] T. Yan, V. Kumar, and D. Ganesan. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In *MobiSys '10: Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 77–90, 2010.

[114] Z. Yang, S. Zhong, and R. N. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *Proceedings of SIAM International Conference on Data Mining*, pages 92–102, 2005.

[115] A. C. Yao. How to generate and exchange secrets. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.

[116] S. Young, D. Kershaw, J. Odell, V. Valtchev, and P. Woodland. *The HTK Book (for HTK Version 3.0)*. Microsoft Corporation, Redmond, WA, USA, 2000.

[117] H. Zhen, H. Wang, and N. Black. Human activity detection in smart home environment with self-adaptive neural networks. In *Proceedings of IEEE International Conference on Networking, Sensing, and Control*, pages 1505–1510, 2008.

[118] G. Zhong and U. Hengartner. A distributed k-anonymity protocol for location privacy. In *IEEE Conference on Pervasive Computing and Communications*, pages 1–10, 2009.

[119] R. Zhu and Z. Zhou. A real-time articulated human motion tracking using tri-axis inertial/magnetic sensors package. *IEEE Transactions on neural systems and rehabilitation engineering*, pages 295–302, June 2004.

Author’s Biography

Raghu Kiran Ganti was born in a tiny town in India and was brought up in Hyderabad. He graduated with a B.Tech. in Computer Science and Engineering from Indian Institute of Technology, Madras in May 2003. He then joined the University of Virginia and started his Ph.D. in Computer Science. He transferred to the University of Illinois, Urbana-Champaign in August 2005 and obtained his M.S. in Computer Science from UIUC in August 2006. He continued on with his Ph.D. in Computer Science at UIUC from August 2006 until August 2010. He is the recipient of the Siebel scholar fellowship for the year of 2010, which is awarded to the top 80 students across the world’s leading graduate schools of Computer Science and Business. He is a member of the IEEE.