StreetSmart: Modeling Vehicle Fuel Consumption with Mobile Phone Sensor Data through a Participatory Sensing Framework

by

Austin Louis Oehlerking

S.B. Mechanical Engineering (2009)
S.B. Management Science (2008)
Massachusetts Institute of Technology

Submitted to the Department of Mechanical Engineering in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2011

Signature of Author.............................. [illegible]
Department of Mechanical Engineering
August 19, 2011

Certified by.............................. [illegible]
Sanjay E. Sarma
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by.............................. [illegible]
David E. Hardt
Chairman, Department Committee on Graduate Theses

(this page left almost blank)

StreetSmart: Modeling Vehicle Fuel Consumption with Mobile Phone Sensor Data through a Participatory Sensing Framework

by

Austin L. Oehlerking

Submitted to the Department of Mechanical Engineering on August 19, in Partial Fulfillment of the Requirements for the Degree of Master of Science

Abstract

Vehicle energy efficiency has become a priority of governments, researchers, and consumers in the wake of rising fuels costs over the last decade. Traditional Internal Combustion Engine (ICE) vehicles are particularly inefficient on high traffic or urban roadways characterized by stop-and-go driving patters. We have developed a novel regression model named StreetSmart that can serve as a transfer function between 4 traffic classification parameters we call "Energy Indices" and the fuel consumption of specific vehicle makes. In formulating the model, we show that average speed, which is the most common metric used to report traffic, is actually inadequate to quantify the impact of traffic conditions on bulk energy consumption. Rather, we use an analysis of traffic microstructure, which is the detailed acceleration profile of individual vehicles on a road segment. Using data logged on OBD-II and smartphone devices from over 600 miles of driving, we have shown that the model is capable of predicting fuel consumption with an average accuracy of over 96% using regression coefficients obtained from the same vehicle make and similar road types. Mean prediction error for all cases ranged from -2.43% to 0.06% while the max prediction error was 7.85%.

We have also developed a framework for the broader StreetSmart System, a participatory sensing network that will be used to crowdsource mass quantities of smartphone accelerometer and GPS data from drivers. We propose a system architecture and discuss problems of distribution, reliability, privacy, and other concerns. Finally, we introduce future applications of StreetSmart, including hybrid vehicle drivetrain power management, electric vehicle range estimation, congestion pricing, and traffic data services.

Thesis Supervisor: Sanjay E. Sarma

Title: Professor, Mechanical Engineering

Acknowledgements

I owe a debt of gratitude to my advisor, Professor Sanjay Sarma, for introducing me to this research project and inviting me to help get components of it off the ground at the Field Intelligence Laboratory (FIL) of MIT. Additionally, I would like to thank him for the years of invaluable guidance he provided while I pursued both my undergraduate and graduate degrees at MIT. Furthermore, I could not have completed this work without the extensivc collaboration of Ajay Deshpande and Stephen Ho on the entire StreetSmart project. Many of our FIL colleagues also provided us with critical feedback and other technical guidance, including Rahul Bhattacharyya, Isaac Ehrenberg, Radu Gogoana, Sumeet Kumar, Joshua Siegel, Ivan Sergeev, Christian Floerkemeier, Edmund Schuster, Benny Drescher, and Dominique Guinard. Additionally, we received substantial input and guidance from many other members of the MIT community, including Amit Goyal and Vikram Mittal.

Finally, and most importantly, I would like to thank my parents, Tom and Ingrid Oehlerking, as well as my sister, Erica Moody, for their unending and loving support of all my pursuits.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Automobile manufacturers and transportation researchers have made vehicle efficiency a priority over the last decade. This trend is the result of consumer demand in the wake of rising fuel prices as well as the United States and European governments' requirements for substantially improved vehicle fuel economies over the next ten years through stricter CAFÉ standards. To improve traditional internal combustion engine (ICE) vehicle mileage, manufacturers are focusing on designing more efficient drivetrains, using advanced materials to reduce weight, and employing sensors and electronic control units (ECU's) to optimize subsystem control. Hybrid vehicles are also gaining prominence as mainstream alternatives to traditional vehicles, coming a long way since the original Prius was introduced by Toyota in Japan in 1997. All of the major manufacturers now have lineups of hybrid vehicles that are gaining market share. Additionally, most large manufacturers as well as a wide array of startups are designing and releasing plug-in hybrid electric vehicles (PHEV) or battery electric vehicles (BEV).

In addition to vehicle design, traffic congestion heavily influences worldwide fuel usage. The US Bureau of Transportation Statistics reports that drivers spent over 4.2 billion hours waiting in traffic during the year 2000 [1]. Since the transportation sector is responsible for 29% of the world's energy consumption, significant traffic congestion can contribute extensively to global greenhouse gas emissions and wasted fuel [2]. Many efforts have been made by researchers to more fully understand the nature of traffic and to explore methods of reducing congestion in urban areas and on highways.

Meanwhile, as transportation technologies continue to improve, the world is being transformed by the overwhelming prevalence of smartphones and mobile technologies that have begun to dominate the marketplace over the last several years. Consumers, developers, and researchers now have access to a variety of useful sensors that open up an array of possible applications. In a transportation context, common smartphone sensors such as accelerometers, gyroscopes, and GPS receivers can provide a detailed profile of individual vehicle motions and locations. Mobile-compatible devices have also been developed to interface with vehicles' OBD-II ports, providing direct vehicle sensor data such as speed, fuel use, and engine statistics [3]. A greater volume of vehicle data is now available than ever before, and this trend will continue as mobile devices become even more ubiquitous. The concept of "participatory sensing", in which many smartphone users contribute contextual sensor data that can be accumulated and used for applications interesting to the group as a whole, has also become a popular idea among researchers and developers along with the rise of mobile technologies.

This thesis explores several applications of mobile and vehicle sensors in the context of transportation. In particular, the characterization of traffic congestion and the quantification of fuel consumption savings is investigated. Additionally, we[1] have developed a framework for a crowdsourcing platform named StreetSmart that can process OBD-II and smartphone sensor data for use in a traffic-themed applications layer. Chapter 1 introduces the automotive and mobile industry trends that make the StreetSmart idea relevant and presents the basic system framework that is developed in this thesis. Chapter 2 analyzes a novel fuel consumption model for ICE vehicles that combines vehicle-specific regression coefficients with real-time traffic parameters

[1] Sanjay Sarma (Professor, MIT Department of Mechanical Engineering)
Ajay Deshpande (Research Scientist, IBM Research)
Stephen Ho (Research Scientist, MIT Department of Mechanical Engineering)

to estimate fuel use. Chapter 3 details StreetSmart, a participatory sensing system for collecting driver smartphone data that can be used for applications such as traffic congestion and fuel use minimization. Chapter 4 discusses a range of other applications of the StreetSmart system, including hybrid vehicle drivetrain optimization, electric vehicle range estimation, congestion taxes, and traffic data reporting. The chapter finishes with a description of future work, challenges, and conclusions.

1.1 Sensors in Road Vehicles

Since 1996, the US Department of Transportation has required all new vehicles to conform to the OBD-II (on board diagnostics) standard, meaning that every new vehicle has an accessible port providing access to a particular range of vehicle sensor data and output [4]. OBD-II is used by technicians to obtain error codes from vehicle ECU's as a diagnostic tool. OBD-II can also be used to measure and log many real time parameters including vehicle speed, engine rpm, throttle position, etc. The vehicle's CAN-bus connects the component ECU's, which control all major subsystems including the powertrain, drive-by-wire functionality, safety systems, and many others. The trend towards electronic sensing and control has been accelerating since the 1980's, when the first CAN chip was produced. New cars typically include over 70 embedded microcontrollers.

Our research centers around the idea of transforming vehicles themselves into useful sensors. One method of obtaining high resolution data from vehicles is to use commercial or custom-designed OBD-II loggers during trips, recording information such as velocity, fuel usage, and GPS coordinates (either via a built-in navigation unit or a separate GPS receiver). After reverse geocoding GPS data and performing the necessary map matching, simple velocity profile and fuel consumption data can be used to construct a detailed view of traffic conditions at any given time, which we refer to as the traffic microstructure.

The downside of logging OBD-II data is that it requires somewhat expensive peripheral devices not typically in the possession of everyday motorists. So, in order to build a comprehensive set of traffic data from the distributed logging of many vehicles, it is critical to find a less expensive method of crowdsourcing. However, the fuel use data from OBD-II logging enables us to create regression models for fuel consumption based on the same acceleration and velocity parameters collected on smartphone sensors.

Motion and location related sensors in smartphones are potentially very useful for transportation applications. Accelerometers can detect a vehicle's motion and can also be used to calculate vehicle speed and position, although with decreasing accuracy. Gyroscopes can be used to resolve the orientation of a smartphone relative to a shifting frame of reference inside of a moving vehicle. GPS can be used for speed measurements as well as accurate location determination. Using a simple reverse geocoding lookup in addition to a map matching algorithm, the location of data segments can be easily determined.

The primary drawback of using smartphone sensors and the phone as a logging tool is the negative effect on the phone's battery life. GPS, in particular, is quite power hungry [5]. So, any sensing scheme to measure the movements of a vehicle would ideally focus on accelerometer measurements with periodic GPS polling to determine street location.

1.2 Sensor Crowdsourcing for Transportation

Participatory sensing is the concept of crowdsourcing a large amount of sensor data from many users who then each benefit from applications based on the dataset. Smartphones make particularly suitable platforms on which to crowdsource sensor data. Multiple research efforts have built and tested such systems, including iCarTel of MIT [6], GreenGPS of UIUC [7], and Mobile Century of UC Berkeley [8]. Kaler et al of MIT CSAIL has also proposed a framework for researchers to collaborate and share data from one larger participatory sensing network [9].

In order to capture a snapshot of the traffic around an urban area at any given time and build reliable applications, a significant number of sensors must be reporting information. Clearly, the most cost-effective method of accomplishing this would be to utilize installed data collection applications on commuting drivers' smartphones. A higher cost method would be to use embedded OBD-II units with mobile capability to collect more detailed vehicle information of commuters. However, this method is not practical for wide distribution because of the high costs and associated privacy concerns of most individuals. A third option would be to install OBD-II loggers in commercial or government vehicles such as taxi cabs and busses, providing both a large amount of useful data as well as a mitigated privacy concern.

1.3 StreetSmart System Overview

The system that we propose in this thesis, StreetSmart, collects data from two parallel streams of crowdsourced data. An illustration of the vehicle logging process is presented in Figure 1.1. The first source is a small but diverse collection of vehicles that have dedicated OBD-II and GPS loggers installed. From these, high resolution fuel consumption, velocity, and GPS data is logged. The data is then uploaded to the StreetSmart server either manually or over the GSM network. Linear regressions based on the 4 "Energy Indices" outlined in Chapter 2 are then run on the data to come up with coefficients that will serve as a transfer function between the Energy Indices of any road and the fuel consumption of that particular type of vehicle on the road. The second data source of StreetSmart is a widely distributed crowdsourced network of smartphones collecting accelerometer and GPS data. The phones upload the data to the StreetSmart server periodically, which then matches the accelerometer values with specific road segments by reverse geocoding the GPS data. The server calculates the 4 Energy Index values for each road segment, which are then stored for use by applications. These numbers are a valuable representation of real-time traffic conditions and are also useful for quantifying the energy consumption of specific vehicles and routes.

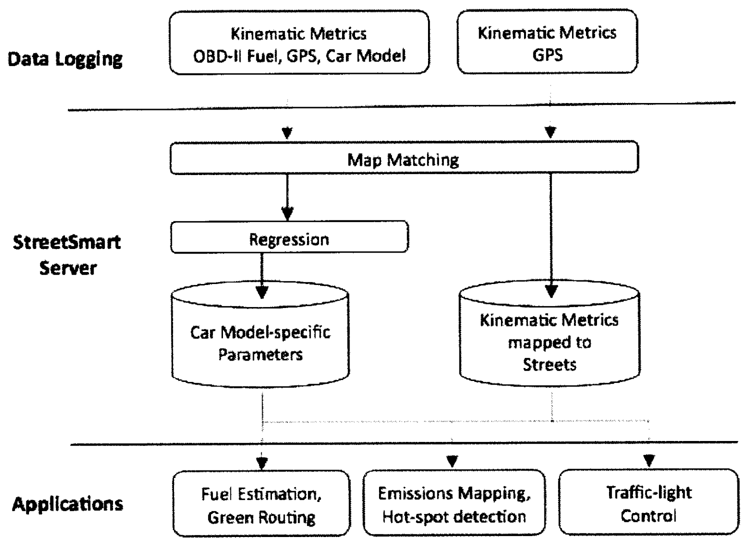

Data Logging
Kinematic Metrics OBD-II Fuel, GPS, Car Model
Kinematic Metrics GPS
Map Matching
StreetSmart Server
Regression
Car Model-specific Parameters
Kinematic Metrics mapped to Streets
Applications
Fuel Estimation, Green Routing
Emissions Mapping, Hot-spot detection
Traffic-light Control

Figure 1.1: StreetSmart Server Data Flow

Chapter 2 ICE Vehicle Fuel Estimation Using StreetSmart Model

As the number of road vehicles commuting into urban areas continues to increase, traffic congestion has become a major concern in terms of air pollution, energy consumption, and logistics. The central goal of the StreetSmart system outlined in this thesis is to provide a framework for processing sensor data from smartphones and vehicles that can help to alleviate these issues. The main component of this framework is a model used to predict vehicle energy consumption based on contextual, real-time measures of road traffic. We have collected and analyzed traffic, road, and vehicle data in order to provide users with the capability of predicting the energy use of specific vehicles along arbitrary routes. We especially aim to improve on fuel use models in the literature, which have generally performed poorly in dense, urban areas or on shorter routes without the advantage of zero-mean error noise cancellation. We have attempted to

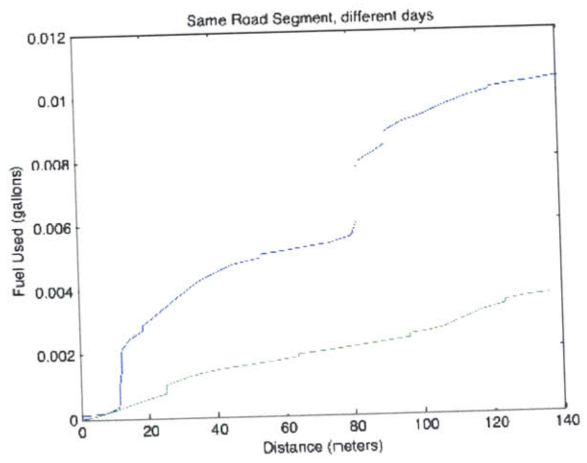


Figure 2.1: Fuel Use of Vehicle on Route with Two Different Traffic Scenarios

link measures of road traffic to vehicle fuel consumption and demonstrate that average speed alone is an inadequate metric to form a complete description of traffic congestion.

We have logged an extensive amount of OBD-II data in a range of vehicle types around the Boston and New York areas. A quick motivation for gathering real time traffic data is provided in Figure 2.1. Two traversals of a road segment are depicted, revealing that the same vehicle consumes 3 times as much fuel in heavy, rush-hour traffic as it does in light traffic. Accurate, up-to-date data can be extremely valuable for the overall energy efficiency of a city's transportation infrastructure if it is used in an intelligent and proactive way. Figure 2.2 depicts the velocity profiles of three separate traversals of the same 1-mile urban road segment containing multiple traffic lights. A significant variation is observed in the driving profiles. In the first, the vehicle moves freely without much stopping. In the second, the vehicle catches multiple traffic lights, while in the third, the vehicle experiences an instance of stop-and-go traffic. In this example, there is a 17% spread between the fuel consumption of the heavy and light fuel consumption scenarios. We require a statistical model to classify these driving patterns that satisfies at least two conditions: (1) it accurately represents the vehicle physics, enabling the estimation of vehicle energy use from driving profiles and (2) it can predict fuel consumption based on driving profile data obtained from readily available sensors in moving vehicles.

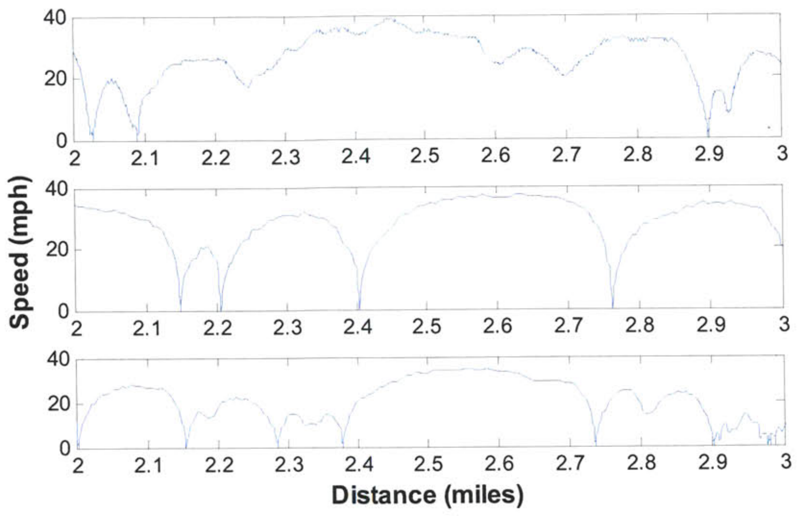


Figure 2.2: Velocity Profiles on 1-Mile Road Segment

In this chapter, we develop and test a fuel consumption model for internal combustion engine (ICE) road vehicles using OBD-II and smartphone data. Section 2.1 provides background information on traffic and fuel consumption estimation methods being used commercially or developed by researchers. In sections 2.2 and 2.3, we develop an initial fuel consumption model based on road vehicle dynamics and prove that "average speed" is not an adequate characterization of traffic congestion when estimating its effect on fuel consumption. In section 2.4, we propose a range of potential "Energy Indices," or parameters that could be useful in a traffic crowdsourcing application that uses mobile phone sensor data as its source. Sections 2.5 and 2.6 describe our OBD-II and smartphone data collection methods and the processing techniques we have developed to use the data in our system. Section 2.7 is a description our process for reconstructing the traffic "microstructure," or a detailed velocity profile of individual vehicle movements, using smartphone sensor data. Finally, Sections 2.8 and 2.9 present the results of our model selection process as well as the model's performance. Additionally, we briefly analyze the persistence of the model parameters as traffic patterns change in time.

2.1 Background and Related Work

Currently, static road sensors such as inductive loop detectors and video cameras are used to measure traffic in many areas by the US Department of Transportation (US DOT). However, these installations can be expensive and are limited to relatively small areas of coverage such as major highways. One of the largest commercial traffic data providers in the United States, NAVTEQ, aggregates data from a wide array of digital traffic sensors, GPS devices, partnerships with local and state governments, and traffic operations centers in major cities [10]. They provide premium traffic data services to customers such as Microsoft's Bing Maps. Another traffic services company, INRIX, was spun out of Microsoft Research and collects real-time crowdsourced GPS information from millions of commercial fleet vehicles as well as from drivers who use the company's smartphone apps. Additionally, INRIX claims to be the only company incorporating a complex set of predictive conditions, such as weather, concerts, sporting events, school schedules, and other local factors into their traffic analytics [11]. Other traffic data companies use cellular signaling data to estimate the average speed of moving vehicles on major roads. AirSage is the most well known of these companies, and it claims to provide accurate traffic information for 85% of the US population through its partnership with major wireless carriers [12].

Researchers have attempted to find more cost effective methods of monitoring real-time traffic congestion. Smartphones have been the key to most of the latest innovations, allowing researchers to use vehicles themselves as traffic sensors by utilizing GPS, WiFi, and other embedded sensor capabilities [13,14,15,6,16,5,17]. The data from these devices can be crowdsourced and used to develop a coherent picture of real-time traffic across a city. As an example, CarTel is a mobile sensor computing platform developed by researchers at MIT CSAIL that uses WiFi signals to analyze vehicle speeds in metropolitan areas [6]. VTrack is a system developed by the same group that uses WiFi and GPS location data to estimate commute times by employing a hidden Markov model-based map matching scheme. They were able to accurately estimate vehicle speed using noisy WiFi data and infrequent GPS sampling [5].

The Mobile Millennium Project at the University of California Berkeley used volunteers' GPS data from smartphones to develop server-side traffic models in the San Francisco area. They have expanded this project to cover travel time estimation, phase transition in traffic, and optimal sensor placement for travel time estimation [17,18,19]. The GreenGPS system, developed at University of Illinois Urbana-Champaign, is an application developed to calculate the most fuel efficient path for vehicles using crowdsourced OBD-II data and trained fuel consumption models [7]. It has been tested primarily in suburban, low traffic conditions, and the project is referred to several times throughout this thesis. Most importantly, GreenGPS focuses on long-term fuel savings and uses estimates of average velocity for road segments in their fuel consumption model. Additional traffic estimation projects are underway for other types of applications. As one example, the authors of [16] use mobile phone sensors to detect potholes, bumps, braking, and honking in order to estimate traffic in developing regions.

While many of the above companies and projects have either dramatically expanded the coverage and accuracy of traffic data or could significantly reduce the costs of doing so in the future, we believe that progress can be accelerated by expanding the definition of traffic to include more parameters than just average speed. For instance, in [20], the authors go beyond a simple speed estimation on roads by carefully observing the spatio-temporal patterns of velocities in order to fully classify states of traffic. In one case, they are able to make a distinction between a vehicle idling at a traffic light and a vehicle in heavy traffic. As the rest of this chapter describes, a dataset featuring more than just average speed could be designed to enable accurate predictions of excess vehicle energy consumption as opposed to just wasted time. This would open the door for a wide array of consumer and city planning applications.

2.2 Vehicle Dynamics and Fuel Consumption Modeling

Most of the projects and techniques noted in the literature can perform relatively well at approximating macroscale traffic delays, but they cannot be used to accurately quantify the fuel consumption of vehicles on a segment-by-segment basis in congested urban driving. We demonstrate in this chapter that an accurate, real-time fuel consumption model for driving requires the examination of the traffic "microstructure." In other words, the detailed profile of a vehicle accelerating and decelerating over the course of a road segment must be analyzed. Average speed or total commute time are not enough to fully characterize traffic conditions.

In order to develop the best model for characterizing traffic microstructure, we begin with an analysis of basic vehicle dynamics and energy requirements. Road vehicle fuel consumption models have been studied in the literature for many years [21,22,23,6,24]. Fuel use is proportional to the instantaneous power demanded by the vehicle integrated over time. A simple vehicle power model accounts for all resistances as well as the dynamic power needed to accelerate or decelerate the vehicle. The instantaneous power (P_e) demanded by a vehicle is represented by Equation 2.1.

$$P_e = \frac{V}{\eta_t}\left(Mgf_r \cos\theta + \frac{1}{2}\rho_a C_D A_f V^2 + P_{acc} + Mg\sin\theta + M\delta \frac{dV}{dt} \right) \qquad \textbf{Equation 2.1}$$

where V is the velocity of the car, η_t is the drivetrain efficiency between the engine and the wheels, M is the vehicle mass, g is gravity, f_r is the coefficient of rolling resistance, θ is the road gradient, ρ_a is the density of the air, C_D is the coefficient of aerodynamic drag, A_f is the frontal area, P_{acc} is the power demand of vehicle accessories, and δ is the mass factor, which accounts for the rotational inertia of the wheels and power plant. The terms represented from left to right are the power demand associated with the vehicle's rolling resistance, aerodynamic drag, accessories (such as air conditioning and lighting), gravitational resistance, and inertial resistance.

Our goal is to create a model using simple kinematic measurements from smartphone sensors that can accurately predict the fuel consumption of a given vehicle. First, we modify Equation 2.1 to eliminate all physical and vehicle dependant constants since regression coefficients from the OBD-II fuel consumption data of each vehicle type will replace them. We use K's in Equation 2.2 to represent the constants.

$$P_e = K_1(\cos\theta + \sin\theta)V + K_2V^3 + K_3\,V\frac{dV}{dt} + K_{acc} \qquad \textbf{Equation 2.2}$$

Assuming that we are driving on flat terrain ($\theta = 0$) and that accessory power is negligible ($K_{acc} = 0$), Equation 2.2 simplifies to:

$$P_e = K_1\,V + K_2V^3 + K_3\,V\frac{dV}{dt} \qquad \textbf{Equation 2.3}$$

If the K_i*'s* are time invariant for particular road segments and vehicles, we can determine the total fuel consumption by integrating the instantaneous power over time. However, we must also recognize that the vehicle is not always moving, especially during times of traffic congestion or when there is a high density of traffic lights. This introduces an additional nonlinearity into the fuel consumption data that could skew the model regression depending on how much time is spent idling. Therefore, we estimate the fuel consumption during moving and idling portions of the trip separately (also see Section 2.8.2).

$$f_{total} = f_{mov} + f_{idle}$$ **Equation 2.4**

$$f_{idle} = \int P_{idle}\, dt = \left(\sum_{seg=1}^{i} t_{idle}\right) k_{idle}$$ **Equation 2.5**

$$\begin{aligned} f_{mov} &= \sum_{seg=1}^{m} \left[k_1 \int_{seg} V dt + k_2 \int_{seg} V^3 dt + k_3 \int_{seg} (Va)\, dt \right] \\ &= \sum_{seg=1}^{m} \left[k_1 \int_{seg} V dt + k_2 \int_{seg} V^3 dt + k_3 \int_{seg} a\, dx \right] \end{aligned}$$ **Equation 2.6**

where *m* is the total number of uninterrupted moving segments (no vehicle stopping) there are on a road and *i* is the number of idling segments in between the moving segments. Unfortunately, traditional ICE vehicles cannot make use of regenerative braking, so Equation 2.6 must be adjusted to account for the fact that vehicle energy is lost in the brakes during negative accelerations, not recovered by the vehicle.

$$f_{mov} = \sum_{seg'=1}^{2m} \left[k_1 \int_{seg'} V dt + k_2 \int_{seg'} V^3 dt + \max(k_3 \int_{seg'} a dx, 0) \right]$$ **Equation 2.7**

where in Equation 2.7, each segment is represented by a period of monotonically increasing vehicle speed or monotonically decreasing vehicle speed. Hence, there are twice as many segments as in Equation 2.6. Alternatively, we can also recognize that the last term in Equation 2.6 equals exactly zero for any segment in which the vehicle begins and ends with zero velocity. By introducing a modulo term, we can produce an expression equivalent to Equation 2.7 in which *k₃* will be exactly halved.

$$f_{mov} = \sum_{seg=1}^{m} \left[k_1 \int V dt + k_2 \int V^3 dt + k_3' \int |a| dx \right]$$
$$where \quad k_3' = \tfrac{1}{2} k_3$$ **Equation 2.8**

In [7], the authors present a formula similar to Equation 2.6, but with the last integral term (inertial resistance) replaced by the square of the average velocity. In terms of data collection, this simplifies the model considerably, since only the average speed of a road segment needs to be known at any given time in order to make a prediction about the overall fuel consumption. However, as we demonstrate in Section 2.3, the average velocity is not a sufficient description of traffic congestion. Even if a vehicle drives with the same average velocity on a road during two separate traversals, its fuel consumption can vary significantly depending on the microstructure of its exact accelerations and decelerations. Additionally, the authors of [7] account for idling time by adding parameters for the number of stoplights and traffic lights to an additional inertial integral. However, especially in city and high traffic driving, the piecewise nonlinearity of the idling term makes such an approach suboptimal. Rather, if idle time can be tracked as a parameter directly, the idling fuel consumption can be predicted with a near 100% accuracy, as is demonstrated in sections 2.8 and 2.8.2.

Our derived fuel consumption model for a vehicle trip is summarized in Equation 2.9:

$$f_{total} = k_i \sum_{seg=1}^{i} t_{i-seg} + \sum_{seg=1}^{m} \left[k_1 \int_{seg} V dt + k_2 \int_{seg} V^3 dt + k_3 \int_{seg} |a| V dt \right]$$ **Equation 2.9**

where k_i is the idling fuel constant, $t_{i\text{-}seg}$ is the idling time of the given segment, and k_1, k_2, and k_3 are the constants for the regression parameters of the moving segments of the trip. Although this equation makes intuitive sense because it accounts for the major vehicle resistances, there are a couple of motivations for adjusting the model further. First, the primary smartphone sensor we would prefer to rely on is the accelerometer. As outlined in [25], accelerometers are far less power hungry sensors than GPS in smartphones. Because of this, we would prefer for GPS to only be used intermittently for map matching and less for direct velocity measurements. Deriving velocity from accelerometer data (see section 2.7) will always introduce some level of error (just as GPS velocity measurement will). The second integral in Equation 2.9 cubes the velocity, which cubes the velocity error as well. This could lead to heteroscadastic errors in our regression, which we will examine more in section 2.8.

So, in finalizing the fuel consumption model, it would be ideal to rely on raw accelerometer and reconstructed velocity data while avoiding terms that will square or cube sensor noise or other errors. While Equation 2.9 represents the ideal vehicle-dynamics-based regression model for StreetSmart fuel consumption estimation, it must be tested against other potential parameters that could simplify the data processing and minimize sensor error. The full array of potential parameters, or "Energy Indices" as we call them, are investigated in section 2.4.

2.3 ADVISOR Simulation – Inadequacy of Average Speed

As described previously, commercial data suppliers and researchers typically correlate traffic congestion to the commute time (equivalent to the average speed) of vehicles on particular roads. All major traffic analytics companies provide an estimated average speed as a measure of congestion on highways and arterial roads [12,26,11]. Many potential applications of traffic data involve estimating the fuel consumption of a vehicle in real-time traffic, such as predicting the most fuel efficient path of a vehicle, determining urban congestion taxes, or reporting the excess energy wasted by a particular traffic jam. However, for applications seeking to quantify the energy impacts of traffic in addition to the average speed, it is significantly more difficult to measure and predict fuel consumption than commute time. The two values can actually diverge significantly.

Intuitively, a lower average speed on a road implies heavier traffic. Traffic is usually associated with additional fuel waste. This result assumes, however, that a traffic jam is significant enough to cause stop-and-go conditions. If a vehicle merely lowers its speed, it will typically consume less fuel due to the decrease in aerodynamic drag. In other words, driving at a steady slow speed is more efficient than stop-and-go driving. So, measuring a vehicle's average speed alone is inadequate to connect specific vehicle driving patterns to fuel consumption – we need a more complete view of the velocity microstructure, which is the detailed picture of a vehicle's accelerations.

In order to simulate minor variations of traffic microstructure in a controlled manner, we use AVL's ADVISOR (Advanced Vehicle Simulator). The tool was originally developed at the US Department of Energy's National Renewable Energy Laboratory before being licensed to AVL [27]. It is built in a Matlab and Simulink environment, simulating vehicle drivetrains and real-world drive cycles in detail. Specific vehicles, including hybrid powertrains, can be modeled in the program and tested against a variety of performance conditions. As a proof of concept, we used OBD-II velocity data from a 2003 Toyota Camry to simulate the vehicle's fuel consumption and compared it to the actual recorded OBD-II fuel consumption. Figure 2.3 demonstrates a clear correlation between the simulated and real-life values. Some of the OBD-II fuel rate peaks are higher than in the simulation, and the OBD-II idling fuel consumption rate is about twice as high as the simulated idling fuel consumption rate. These differences are likely the result of unknown variations in the vehicle's drivetrain that were not incorporated into the default Toyota Camry simulation model.

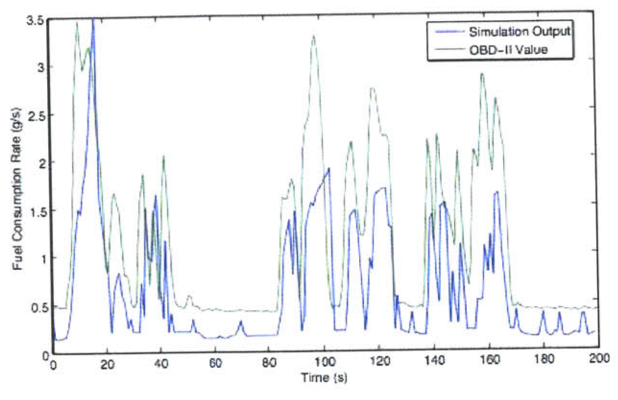


Figure 2.3: Instantaneous Fuel Consumption, Simulated vs. Measured

In order to simulate the effect of traffic microstructure on fuel consumption, we created a series of sawtooth driving profiles, each of which had the same average speed of 15 miles per hour. All runs covered exactly 10 miles over a time period of 2,500 seconds. In each simulation, we increased the number of sawtooths by one, simulating slightly heavier stop-start traffic conditions. In the limiting case of about 100 sawtooths, the vehicle was no longer able to achieve the requested velocity profile without deviation. In Figure 2.4, an example drive cycle is presented with 8 sawtooths. The vehicle is always accelerating or decelerating and is never allowed to idle or maintain a constant velocity. The key factor that changes is the rate of acceleration and deceleration needed in order to achieve a certain number of fluctuations over the given path.

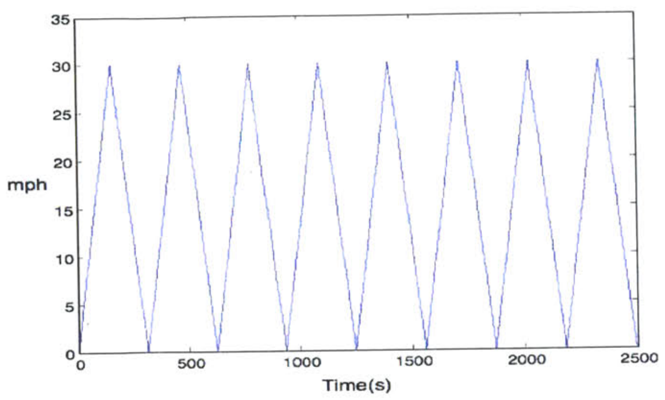


Figure 2.4: Sawtooth Velocity Profile with 8 Ramps

As can been seen in Figure 2.5, as the number of accelerations and decelerations increases, so does the fuel consumption of the vehicle. This would be consistent with our expectation that a more volatile or noisy traffic microstructure would lead to higher fuel consumption. The largest fuel consumption rate (at high stop-and-go frequencies) in Figure 2.5 is over 50% higher than the lowest rate (at a constant velocity). This is a conclusive demonstration that average speed alone is not an adequate description of traffic congestion if the aim is to predict the impact on fuel consumption. A detailed view of the traffic microstructure is needed to make that assessment.

Finally, we used ADVISOR to refine our estimation of the required OBD-II sampling rate of velocity data. In a real world deployment of StreetSmart, regression models would need to be constantly refined for all vehicle types as more volunteer OBD-II data becomes available. Additionally, unless on-board calculations could be made with the smartphone or OBD-II data, the entire microstructure dataset would need to be transmitted to the StreetSmart server for data processing (see section 2.6). Minimizing data transmission costs is important for any user transmitting a large number of sensor readings over time.

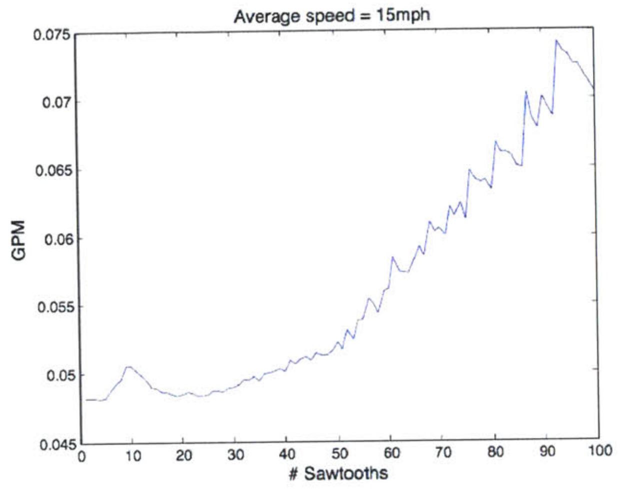


Figure 2.5: Fuel Consumption (Gallons per Mile) for Increasing Traffic

In order to determine the minimum velocity sampling rate, we downsampled three OBD-II data collection runs from Memorial Drive in Cambridge, MA, reducing the original sampling rate from a little over 1 Hz to between 1Hz - 0.02 Hz. The downsampled velocity profiles were then simulated in ADVISOR and the resulting errors in fuel consumption prediction are shown in Figure 2.6. For each of the data runs, it is clear that a sampling rate of at least 0.3-0.5 Hz is needed in order to keep error rates below 5%.

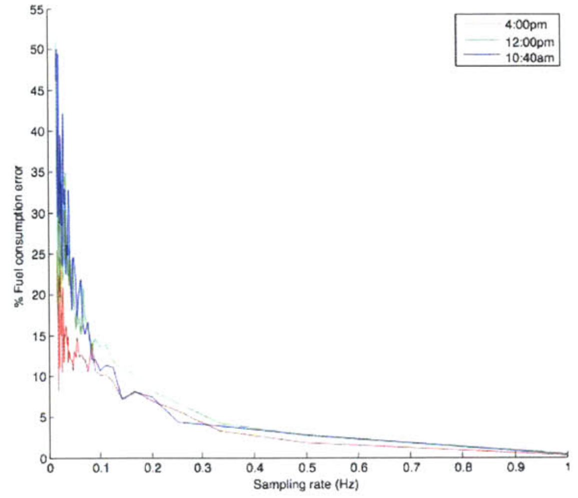


Figure 2.6: Effect of Sample Rate on Fuel Estimation Accuracy

2.4 Fuel Estimation Model – Energy Indices

In Section 2.2, we derived a fuel consumption model for an ICE vehicle based upon vehicle dynamics and power requirements. However, the model represented in Equation 2.9 has a couple of drawbacks if the data is being collected on smartphone accelerometers. Reconstructing velocity values from acceleration measurements introduces error. The error is cubed for the velocity cubed term in Equation 2.9, leading to poorer linear regression accuracy. If we want to reduce the velocity errors, we can use direct GPS velocity outputs. However, this approach has battery life [25,28] as well as accuracy disadvantages (see section 2.7).

By taking into account Equation 2.9 as well as real-world considerations about traffic, vehicle motion, and available smartphone sensors, we present a set of metrics that are considered for use in the vehicle fuel consumption model. We collectively label these terms "Energy Indices" since they represent terms that would seem to correlate with energy use in vehicles. In sections 2.8 and 2.9, we evaluate the Energy Indices with the available data for accuracy and persistence, selecting a final fuel consumption model.

Table 2.1: Energy Indices Considered for Model Selection

	Parameter	**Parameter Description**
1	T_{idle}	= (Total Vehicle Idling Time on Road Segment)
2	T_{moving}	= (Total Vehicle Moving Time on Road Segment)
3	$\int Vdt$	= (Segment Length)
4	$\int Vcos(\theta)dt$	α (Energy to Overcome Rolling Resistance)
5	$\int Vsin(\theta)dt$	α (Energy to Make Elevation Changes)
6	$\int V^2dt$	α (Drivetrain Resistance) [22]
7	$\int V^3dt$	α (Energy Used to Overcome Aerodynamic Drag)
8	$\int \lvert a\rvert dt = \int \lvert dV\rvert = \sum \lvert \Delta V\rvert$	= (Sum of Velocity Changes on Segment)
9	$\int \lvert VdV\rvert = \int \lvert a\rvert Vdt = \int \lvert a\rvert dx$	α (Total Positive Energy used to accelerate vehicle)

Table 2.1 lists and described all of the possible Energy Indices. These are evaluated as regression parameters for vehicle fuel consumption in Section 2.9. Parameters 4, 5, 7, and 9 were found in Equation 2.1, with each term representing the energy that must be expended by the drivetrain to overcome one of the major vehicle resistances. However, due to estimation errors of the velocity, these parameters alone may not make the best overall model for vehicle energy consumption.

Parameters 1, 2, 3, 6, and 8 are potential common sense parameters that would intuitively correlate either with fuel consumption or traffic congestion directly. The idling and moving times of the vehicle would be ideal model components because they can be recorded accurately and easily from smartphone accelerometer data. Additionally, the idling fuel consumption rate tends to remain nearly constant, making the idling time a great predictor of idling fuel consumption. Vehicle moving time is like a more useful measure of the average vehicle speed of a road segment but with the nonlinearity of idling time removed. The length of a segment is a constant value which would naturally affect fuel consumption. Finally, the integral of the absolute value of acceleration as a function of time is equivalent to summing the absolute value of all of the velocity changes of a road segment. This value could be another way of representing stop-and-go traffic conditions as an alternative to parameter 9, which has a more direct proportionality to energy.

The actual StreetSmart system employs a fuel consumption model that uses a subset of the parameters listed in Table 2.1. The parameters are regressed on velocity profile and fuel consumption data for many different types of vehicles on selected road segments. Each vehicle then has stored coefficients from the regressions which can be used to predict the fuel consumption of that type of vehicle on particular roads for which real-time traffic microstructure data exists. The microstructure data for a road is crowdsourced by driver smartphones that collect accelerometer and GPS data and condense it into the selected parameters for each road segment before uploading it to the server, as is outlined in more detail in Chapter 3.

2.5 Data Collection

The StreetSmart system relies upon a fuel consumption regression model based on logged OBD-II speed, fuel rate, and GPS location data from vehicles. In an actual production version of

StreetSmart, some small percentage of users would volunteer to use a mobile-enabled OBD-II logging tool such as presented by Siegel in [3]. This would allow detailed models to be trained by the StreetSmart system matching specific vehicle and road types to fuel consumption patterns for given traffic microstructures. To test the model, we have logged over 600 miles of driving data in 10 vehicles. The 5 driving routes consisted of both urban and suburban environments during both heavy and light traffic periods. The data is primarily from the Boston and Cambridge areas with some also from a daily commute in Ossining, NY.

There are multiple commercial OBD-II logging units available on the market, such as the Autotap Reader [29] and the AutoXRay Ez-scan [30]. We use the Auterra DashDyno (Figure 2.7) because of its synchronized GPS capability and its derivation of real-time fuel rate from vehicle sensors [31]. It can log an arbitrary number of real-time vehicle parameters on a removable SD memory card. Its fastest sampling rate is determined by the vehicle, with newer models typically recording data at up to 2 Hz.

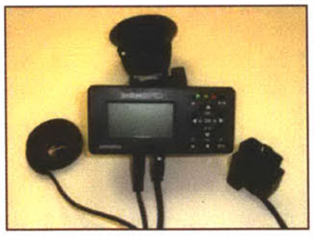


Figure 2.7: Auterra DashDyno (center), GPS Unit (left), and OBD-II Connector (right)

Additionally, we have used a variety of accelerometers to record vehicle motion (Figure 2.8). The Nokia N95 was the primary smartphone used. On Android and iPhone platforms, accelerometer sampling occurs in an event-driven manner, making a fixed sampling rate difficult to achieve for logging purposes (at the time of this writing). On average, an accelerometer sampling rate of 30 Hz was achieved from the smartphones (the Nokia N95 allowed fixed sampling). We also included a GC Data Concepts precision accelerometer on many of our data collection runs, providing a sampling rate of 80 Hz and an improved sensitivity over the smartphone sensors.

Figure 2.8: Smartphones and a Precision Accelerometer

Table 2.2: Vehicles Used

Vehicle	**Miles Logged**	**Routes [See Table 2.3]**
Toyota Camry 2003	200	MD, MA, CA, SS, OH
Toyota Highlander	18	MD
Toyota Camry 1998	40	OH, SS, MD
BMW 328xi	42	MD
Volvo S40 (1)	66	MD
Volvo S40 (2)	30	MD
Mazda 3	20	MD
Kia Soul	26	BM
Honda Insight	23	BM
Honda CR-V	50	MA
Toyota Prius	30	MA
Hyundai Sonata	25	MD

We collected data from a variety of vehicles, primarily rented from ZipCar [32] in the Boston region. With the exception of the hybrid Toyota Prius and Honda Insight, all of the vehicles were traditional internal combustion engine (ICE) models. The vehicle modes, routes, and associated miles logged are summarized in Table 2.2. The routes selected represented both urban and suburban environments in both heavy and light traffic scenarios. Route locations, speed limits, and traffic conditions are summarized in Table 2.3. Round trip maps of three of the routes are presented in Figure 2.9 as well.

The StreetSmart System relies upon vehicle fuel consumption following predictable patterns for a given street classification, driving condition, vehicle type, and traffic microstructure. By demonstrating a persistence in the StreetSmart Energy Indices over various segments and times, the full model can be validated (see Section 2.9 for more detail). We measured road type persistence by comparing derived Energy Indices across different road segments. We also collected several data runs of two vehicles following the same route with different time separation intervals and evaluating the resulting Energy Indices to determine the persistence of traffic microstructures over time.

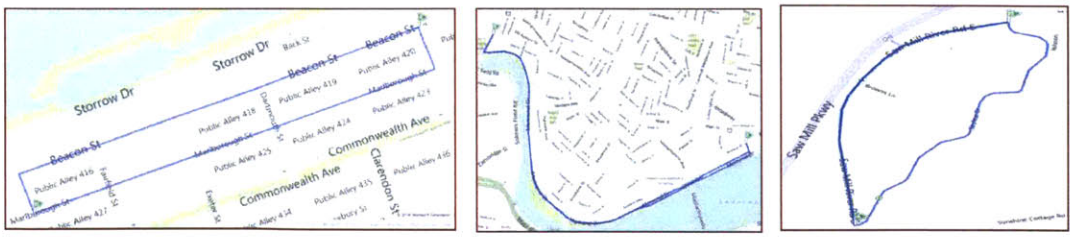


Figure 2.9: Streets: Beacon-Marlborough, Memorial Drive, and Skyline Drive–Sawmill

Table 2.3: Road Routes

Street Names	**Speed Limit (mph)**	**Traffic Encountered**
Memorial Dr (MD)	35	Medium, Heavy
Beacon St (BM)	30	Light, Medium, Heavy
Massachusetts Ave (MA)	30	Heavy
Skyline Drive – Sawmill Rd (SS)	30-45	Light, Medium
Ossining-Hawthorne, NY (OH)	30-55	Light, Medium

2.6 Data Processing: Sensor Output to Energy Indices

Once the data is collected, there are multiple processing steps that must occur before integrating it into the StreetSmart System. Chapter 3 provides a more in depth view of the overall structure of StreetSmart as well as its processes and components. This section describes the actual algorithms necessary for processing the data before it can be used. As has been described, the StreetSmart System relies on two data sources. First, OBD-II logging of a variety of vehicle models and road types provides velocity profile and fuel consumption data that can be used to train the fuel consumption model. This data allows us to obtain vehicle-specific and road type-specific coefficients for Energy Indices. Second, real-time crowdsourced smartphone accelerometer and GPS data is used to create a map of traffic microstructures across a road network. The combined data of these two sources can be used to predict the fuel consumption or find the most fuel efficient route of a vehicle about to make a trip, for instance. Depending on if OBD-II or smartphone data is being collected, a slightly different set of data processing steps must be followed, which is outlined in Table 2.4.

Table 2.4: Data Processing Steps

Steps	OBD-II	Smartphone
(1) Snap GPS coordinates to road grid	Yes	Yes
(2) Reverse Geocode GPS coordinates	Yes	Yes
(3) Reconstruct Velocity Profile	No	Yes
(4) Determine Segment Splits	Yes	Yes
(5) Split Segment Data into Idling and Moving Components	Yes	Yes
(6) Calculate and Store Energy Indices for Each Segment	Yes	Yes
(7) Regress Energy Indices on Fuel Consumption to Train Vehicle Model	Yes	No

Step (1) has already been addressed by a prior data collection effort in our lab and is outlined in [33]. Step (3) is outlined in Section 2.7, and Steps (6) and (7) are covered in Section 2.9.

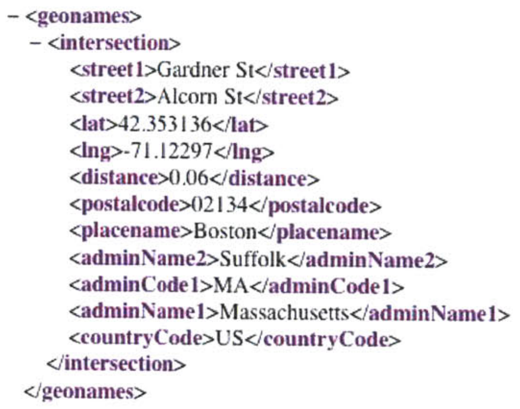
2.6.1 Reverse Geocoding GPS Coordinates

Reverse geocoding is the process of translating recorded GPS coordinates into useful map data. There are a variety of online reverse geocoding services available, but we made use of the free website GeoNames.org [34], which allows up to 5,000 queries per hour and 30,000 per day on its geographical database. To reverse geocode an entire dataset in Matlab, we simply make a series of requests using a URL with the GPS data formatted as below:

URL Request: http://ws.geonames.org/findNearestIntersection?lat=42.3534&lng=-71.1224

GeoNames.org returns an XML file that can be easily parsed in Matlab. The returned file for the request above is depicted in Figure 2.10. The Matlab reverse geocoding script is found in

Appendix A.1, including error checking for various fault conditions and other adjustments so that the scripts can be run overnight on large amounts of data.



```
- <geonames>
  - <intersection>
      <street1>Gardner St</street1>
      <street2>Alcorn St</street2>
      <lat>42.353136</lat>
      <lng>-71.12297</lng>
      <distance>0.06</distance>
      <postalcode>02134</postalcode>
      <placename>Boston</placename>
      <adminName2>Suffolk</adminName2>
      <adminCode1>MA</adminCode1>
      <adminName1>Massachusetts</adminName1>
      <countryCode>US</countryCode>
    </intersection>
  </geonames>
```

Figure 2.10: XML Data of Reverse Geocoding Request [34]

2.6.2 Determining Segment Splits

After reverse geocoding is finished and a complete velocity profile is available, the segment divisions must be determined. Several variations of segment splitting are possible. In the GreenGPS system [7], roads are split into 1 mile segments, after which the model is trained by regression and the fuel consumption on other 1 mile segments is estimated. In an actual functioning system, however, it is necessary to split up a road network in a manner that will allow routes with turns and intersections to be evaluated while "cutting off" as little data as possible. Using fixed distance increments places the edges of segments in arbitrary locations. In an urban setting, traffic lights are frequently spaced very closely, often less than 1 mile apart. Dividing road segments by traffic light intersections, then, makes more sense than picking a fixed length increment. Figure 2.11 is a map of Memorial Drive in Cambridge, MA, with the traffic light and no-stoplight intersections (for a vehicle traversing Memorial Drive) indicated.

At this point in the process, the StreetSmart system receives reverse geocoded location data either from a crowdsourced smartphone accelerometer or an OBD-II user. Between River Street (Point 1 in Figure 2.11) and JF Kennedy Street (Point 9 in Figure 2.11), the reverse geocoded GPS data indicates that the vehicle crosses 7 intersections. A further examination of the map, however, reveals that only 2 of the intersections have traffic lights while 6 of them are "no-stoplight" intersections for vehicles on Memorial Drive. Additionally, the reverse geocoding process falsely detects one intersection between Plympton and JF Kennedy St.

For a driver on this arterial road, the Energy Indices would be most useful for the 3 traffic-light-to-traffic-light road segments. Since the entire route is only 0.8 miles long and vehicles are unlikely to vary in speed significantly between the intersections except in high traffic, it makes sense to split the data up into these 3 sections.

Figure 2.12 displays the output of the reverse geocoding "Distance to Nearest Intersection" parameter as a function of drive time as well as vehicle speed. In the upper plot, the blue region represents the "River Street – Western Ave" segment, the red region the "Western Ave – Dewolfe St" segment, and the green region the "Dewolfe St – JFK St" segment. Matlab code automatically splits the data into these segments by accessing a list of predefined traffic light intersections and finding the index of the minimum value for the distance variable of these intersections. Other intersections are ignored despite producing local minimums in the distance to intersection variable as seen for points 3, 4, 5, 7, and 8 in Figure 2.12. For this example run, the vehicle stops at the traffic light at each of the three intersections and has a period of idling followed by an acceleration to some maximum speed (about 30 mph in each case).

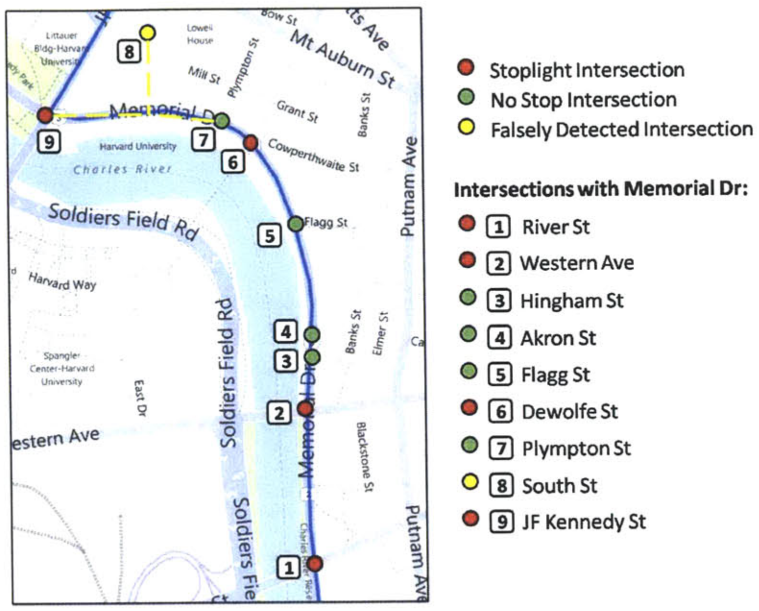


Figure 2.11: Map Showing Derived Intersections from Reverse Geocoding

Since these segments are being divided so that the Energy Index parameters of each can be calculated and recorded, it is important to note the influence of "edge effects." For instance, when the vehicle stops for a red light, it idles some distance behind the actual intersection. As it accelerates through the green light, it has some speed V_{int} through the intersection. As is apparent in Figure 2.12, V_{int} can vary based upon how much time a vehicle has to accelerate from its stopping position at an intersection. Each segment, therefore, starts off with a velocity dependent upon the queue length at the stoplight of the previous segment. In general, we attempt to make the Energy Index data as independent as possible for each segment. Referring to the Energy Index list in Table 2.1, changing the value of V_{int} for a segment may impact parameters 2, 6, 7, 8, and 9.

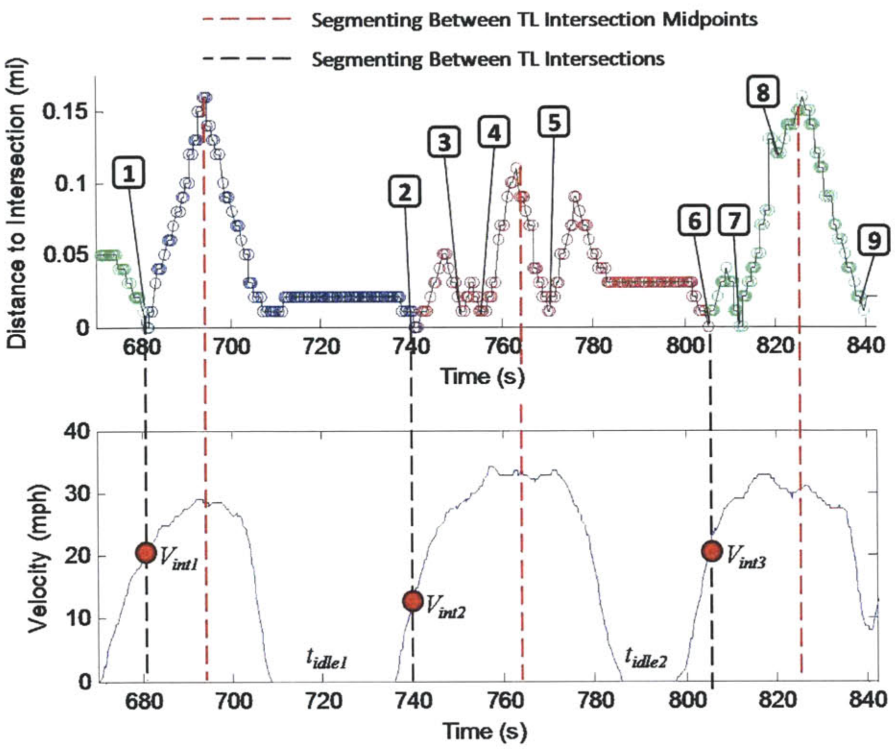


Figure 2.12: Dividing Roads by the Distance-to-Intersection Variable

Alternatively, we can choose to segment the road network from the midpoints between intersections instead of between the intersections themselves. This approach would allow us to account for undesirable edge effects that occur in the intersections themselves. All of the intersection uncertainty would be captured within the Energy Indices of *a single segment*. The algorithm for this technique is suggested by the red vertical lines in Figure 2.12. The end points (intersection nodes) of a segment are found by a search of the 'distance-to-intersection' variable. In the figure, this approach would clearly work for the first and the third segments. The second segment, though, has many 'no-stoplight' intersections that make finding a repeatable, accurate midpoint location by reverse geocoding difficult. To robustly implement this method, it might be necessary to geotag the midpoints of all of the major intersections that are included in a road network.

Figure 2.13 depicts the direct intersection-to-intersection method of covering a road network with StreetSmart Energy Indices. Each 4-way intersection node has 4 unique Energy Indices associated with it. In order to establish a picture of traffic congestion along a route, the Energy Indices of path of intersection nodes can simply be applied to the fuel consumption formula and summed. Figure 2.14 applies the same 4 branch per intersection approach to the segmenting method of connecting intersection midpoints. In this case, all of the traffic dynamics of the intersections are captured by the Energy Indices instead of being split between 2 different Indices.

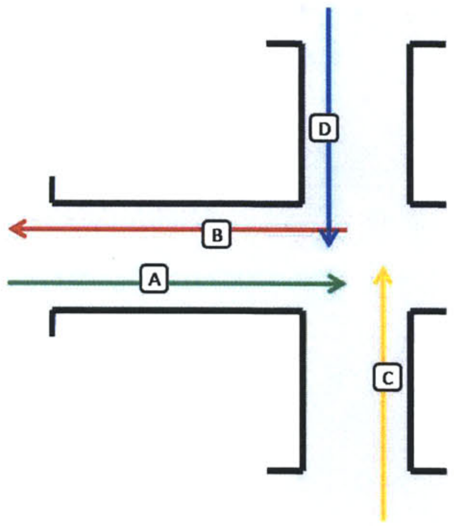


Figure 2.13: Segmenting Method - Connect Major Intersections

Unfortunately, the intersection midpoint method introduces a problem when left or right turns are required. Instead of having segments that can be quickly summed, there is no easy way to establish the Energy Indices around the turning corner except by splitting the segments again or by adding addition segments for each node. In Figure 2.15, the number of branches connecting each 4-street all-way intersection node is tripled from 4 to 12. An additional branch is added for each type of available turn.

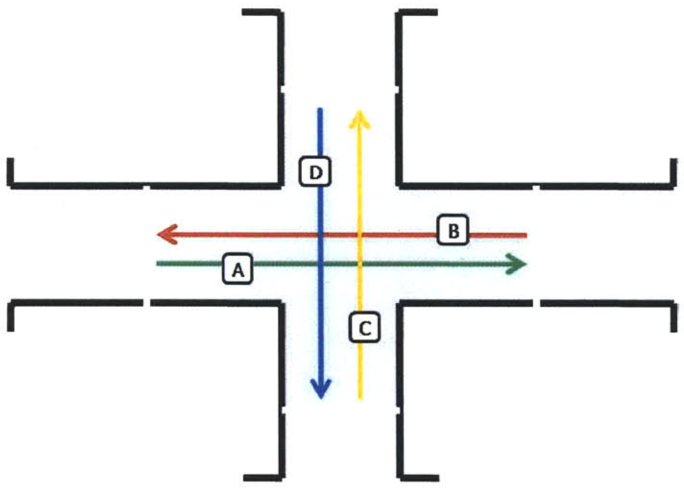


Figure 2.14: Segmenting Method – Connect Major Intersection Midpoints

The advantage of this method is that particular traffic dynamics specific to left and right turns will be separated from the dynamics of vehicles that go straight through an intersection. It is well established that vehicles consume less fuel on average when making right-hand turns versus left-hand turns [35]. Therefore, connecting intersection midpoints and including turns has the potential to form the most complete view of traffic congestion on a road network. Unfortunately, since the number of branches in the network has tripled, this approach is at a major disadvantage when only sparse data is available. If crowdsourcing is the primary data collection mechanism of StreetSmart, this segmenting approach relies on having a very large participation rate in order to get adequate coverage.

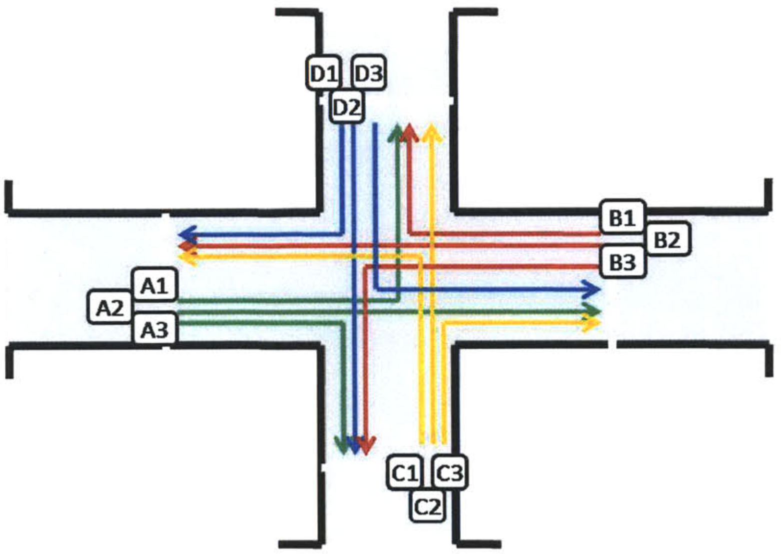


Figure 2.15: Segmenting Method – Connect Major Intersection Midpoints Including Turns

2.6.3 Split Idling and Moving Components

Once particular segments are identified, the last step prior to calculating the Energy Indices is to split up the velocity data into idling and moving components. The Matlab script for this process scans an entire road segment and records the time array indices for the beginning and end of idling periods. The index values are then stored in the data structure of the run for evaluation. Figure 2.16 is an illustration of the resulting components of the idling/moving split algorithm for a section of Memorial Drive. The full code of this algorithm can be found in Appendix A.2.

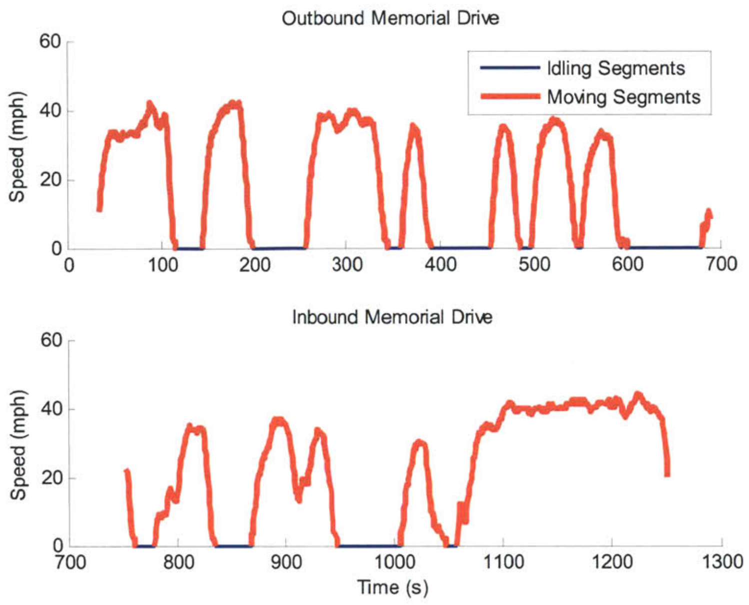


Figure 2.16: Idling and Moving Sections of a Memorial Drive Commute

2.7 Reconstructing Traffic Microstructure: GPS and Accelerometer Data

In order to calculate the Energy Indices for road segments of crowdsourced smartphone data, a complete velocity profile must be logged with GPS data or reconstructed using accelerometer data. We have already established that logging continuous GPS data incurs a steep energy penalty on the phone [36]. As outlined in Section 2.3, we must collect data at a rate of at least 0.3 Hz to 0.5 Hz in order to avoid significant downsampling errors. Some example GPS velocity data is compared to OBD-II data in Figure 2.17. A varying 2-3 second lag of the GPS data behind the OBD-II data is clearly distinguishable. Even with a shift correction, the match would not be exact because of the changing magnitude of the lag. Additionally, in most of the GPS data that we have recorded, there are random outliers in both the position and velocity data that are the result of noise or sensor error. For instance, the single data point dip in velocity in the second peak in Figure 2.17 is clearly an outlier. These points could be automatically detected by a simple script that would detect outliers based on the implied acceleration and remove them as needed.

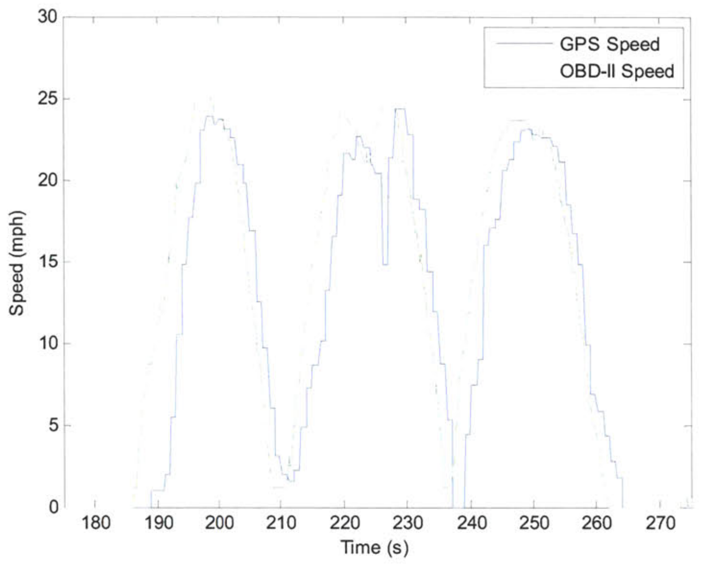


Figure 2.17: GPS vs OBD-II Recorded Speed

Accelerometer measurements have a lower energy penalty than GPS and can also produce accurate results with an adequate sampling rate. Unfortunately, raw accelerometer data is affected by drift errors, random biases, noise, and nonlinearities [37]. Industrial applications have made use of Kalman filters, finite impulse response filters, and the frequency domain integration approach to reduce dead reckoning errors associated with integrated accelerometer data [38]. In order to correct for this, we have worked out methods to reconstruct velocity profiles from smartphone accelerometer data that are specific to automobile driving. In particular, our method makes use of the following observations:

(1) Periods of constant acceleration correspond to zero acceleration

(2) Periods of zero acceleration correspond to zero velocity

(3) Infrequent GPS sampling can provide a scaling factor for reconstructed velocity

(4) Phone orientation can be determined during periods of vehicle idling

The first two notes have a grounding in traditional inertial navigation techniques. For instance, it is similar to the zero velocity update procedure (ZUPT), which relies upon periodic stopping to prolong the accuracy of inertial position calculations [39]. Our first point, however, goes beyond the basics of this well established technique. In a moving vehicle, it is unlikely that a driver could maintain a perfectly steady velocity (and hence zero acceleration) at any speed except for zero. The only exception might be with cruise control activated on an open highway. Therefore, we can correct for drift errors by noting that any constant non-zero acceleration period should be adjusted to zero acceleration.

In order to determine the vehicle's coordinates for reverse geocoding, some GPS sampling will be required. After employing the enhanced-ZUPT technique to correct the accelerometer output, we have found that the reconstructed velocity can still be off from actual values by as much as 50%. If the vehicle is travelling on a straight road, the distance between the GPS measurements can be used to formulate a scaling factor for the velocity, as is calculated in Equation 2.10:

$$V_{scaled} = \left(\frac{dist\ (GPS(t_0), GPS(t_1))}{\int_{t_0}^{t_1} V_{accel}\, dt} \right) \bullet V_{accel} \qquad \textbf{Equation 2.10}$$

where V_{accel} is the reconstructed velocity profile from the ZUPT-corrected accelerometer data. In 13 datasets for a vehicle moving on a straight road, GPS scaling decreased the RMS error of our velocity reconstruction by 54%.

In all of our data collection, the phone was secured with the y-axis parallel to the longitudinal direction of motion of the vehicle. However, in an arbitrary crowdsourcing application, the orientation of the user's phone would have to be determined in order to make use of the accelerometer data. Our technique for accomplishing this is to wait for periods of idling (as determined by steady accelerometer measurements) in order to determine the direction of gravity. After this, the components of gravity can be normalized out of the system and the x-y plane orientation of the phone can be determined. The fact that the car accelerates mostly in the longitudinal direction is used to determine the y-direction. The only inaccuracies arise when the vehicle has significant lateral accelerations on a winding road or during an intersection turn. However, these errors are corrected each time the vehicle comes to rest and accelerates from rest again.

2.8 Logging Results and Model Selection

As discussed in Section 2.4, we separated the real-time OBD-II fuel consumption data into idling and moving components. Idling segments typically demonstrated a near constant fuel consumption rate. Figure 2.18 shows the idling fuel consumption of a 2003 Toyota Camry as well as a Volvo S40.

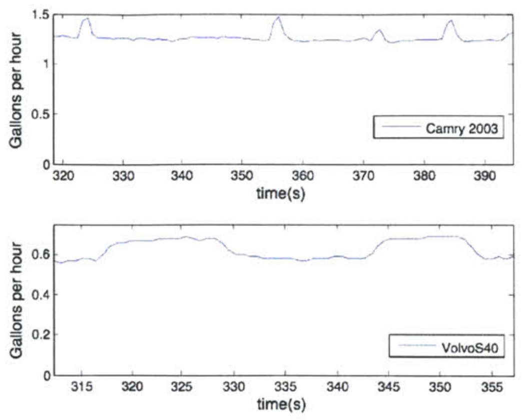


Figure 2.18: Idling Fuel Consumption Rates: Toyota Camry (top), Volvo S40 (bottom)

The idling segments of four vehicle models are plotted in Figure 2.19. In each case, a clearly linear trend between idling time and fuel consumption is observed. Interestingly, the Volvo S40 has a noticeably lower idling fuel rate than the other three models, which all maintain idling fuel rates of about 1.3 gallons per hour.

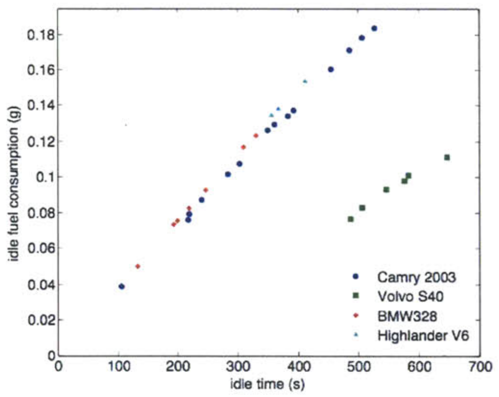


Figure 2.19: Idle Fuel Consumption of 4 Vehicle Models

In Figure 2.20, fuel consumption is graphed as a function of all of the microstructure parameters discussed in Section 2.4. Each plot demonstrates a positive, nearly linear correlation between the parameter and the vehicle fuel consumption with the possible exception of the *integral of acceleration dt.* As predicted, the integrals of velocity squared and velocity cubed exhibit clear heteroscedasticity, making them undesirable candidates for linear regression. This is due to the fact that each is reconstructed from raw OBD-II velocity data. Any nonlinearities, noise, or errors in this data when plotted against fuel consumption is squared and cubed in the last two plots, respectively. This error magnification affects larger values more, resulting in the heteroscedastic error. Commute time and segment length seem to have the smallest error distribution in their linear trends.

Before regressing the parameters, a correlation matrix of the Energy Indices is presented in Table 2.5. Many of the parameters are highly correlated with each other, which is not surprising. Since velocity squared and velocity cubed are derived from velocity, the data will follow the exact same trends. Moving time is a wild card and does not correlate especially well with any of the parameters except for fuel consumption, which is actually ideal for the regression. The two inertial parameters, the *integral of acceleration dt* and the *integral of acceleration dx*, do correlate with each other somewhat, but the latter term correlates more strongly with fuel consumption. This is not surprising, since the *integral of acceleration dx* is proportional to inertial energy costs in the vehicle physics model.

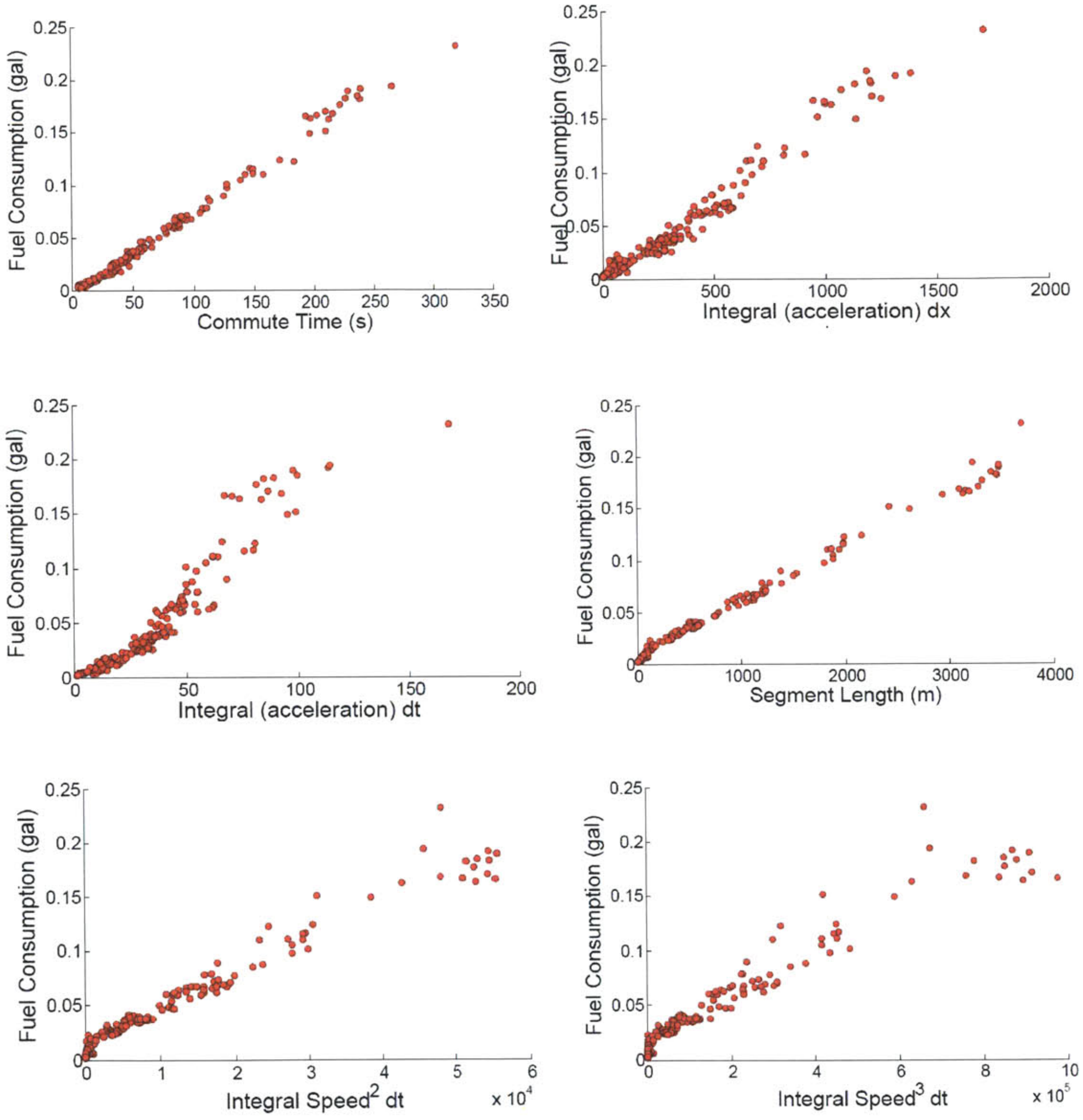


Figure 2.20: Energy Index Parameters vs. Non-Idling Fuel Consumption, 2003 Toyota Camry

	$\int Vdt$	$\int V^2dt$	$\int V^3dt$	T_{moving}	$\int \lvert a \rvert dt$	$\int \lvert a \rvert dx$	FC_{moving}
$\int Vdt$	1.000						
$\int V^2dt$	0.984	1.000					
$\int V^3dt$	0.959	0.993	1.000				
T_{moving}	0.795	0.692	0.639	1.000			
$\int \lvert a \rvert dt$	0.362	0.249	0.203	0.747	1.000		
$\int \lvert a \rvert dx$	0.849	0.803	0.771	0.801	0.713	1.000	
FC_{moving}	**0.930**	**0.864**	**0.822**	**0.948**	**0.631**	**0.887**	**1.000**

Table 2.5: Correlation Matrix for Non-Idling Energy Indices

In order to actually select a finalized model, we conducted an iterative evaluation of different parameter combinations to see how each potential model performs. We have focused on data taken from the Beacon Street – Marlborough Street loop with a Kia Soul (2.0 L 4-cyl. engine, 4-speed automatic, FWD, 26 city/31 hwy mpg). The model evaluation process is described by the following steps:

(1) Split data into two segments

(2) Filter out idling time from each segment so that 'moving' parameters and 'idling' parameters can be regressed separately

(3) Determine linear regression coefficients for each segment using all possible combinations of model parameters

(4) Sort the regression results in descending order by mean error percentage of regression residuals.

(5) Using the regression coefficients determined for each data segment, predict fuel consumption on the opposite segment and evaluate the mean error and error distributions of the estimations.

2.8.1 Iterative Regressions

The Beacon Street – Marlborough Street route is depicted in detail in Figure 2.21, showing the starting and ending locations of the segments (red dots) as well as all traffic lights in between. This route is located in the Back Bay region of Boston, and data was collected at both rush hour and light traffic times. There were a total of 19 data points on each segment.

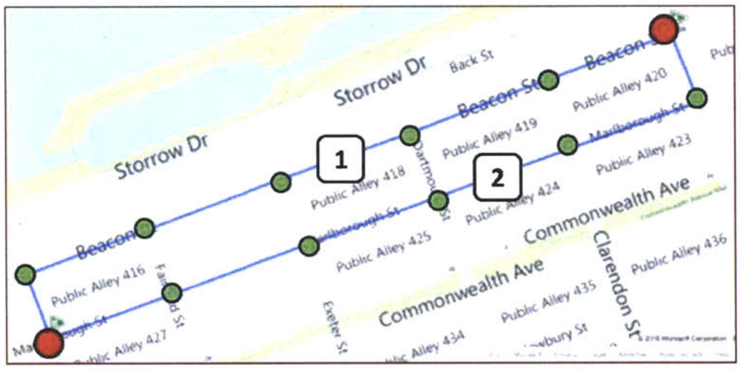


Figure 2.21: Annotated Route Map: Beacon - Marlborough

The various Energy Indices can be tested in 63 independent combinations. Results are depicted in Figure 2.22 and Figure 2.23, representing regressions on Beacon Street and Marlborough Street, respectively. The left bar plot in each of these figures sorts the mean accuracy of the regression in descending order for all of the combinations. Stacked on the accuracy values is the standard deviation of the residuals, indicating how spread out the measurements are from the linear regression fit values. The majority of the regression attempts (63% in Figure 2.22 and 89% in Figure 2.23) achieve greater than a 95% mean accuracy, meaning that there may be several or many combinations of Energy Indices that will be able to adequately measure and predict fuel consumption.

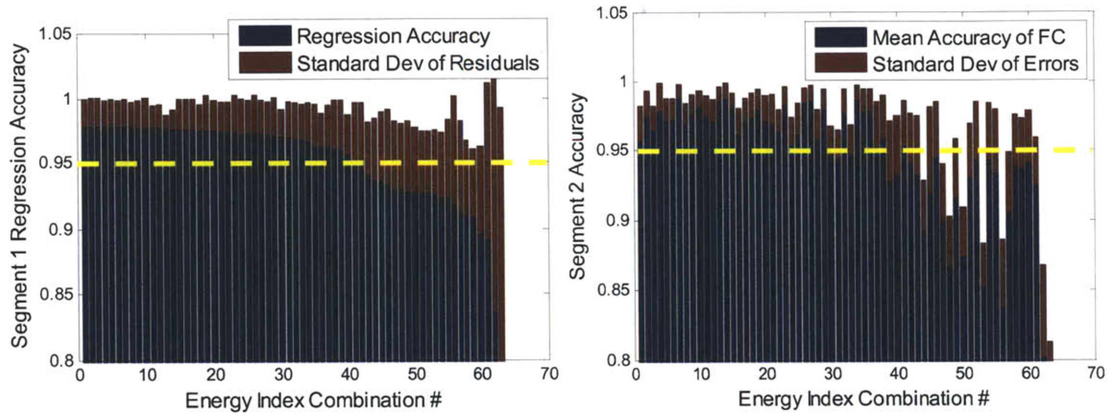


Figure 2.22: Beacon Street Regression with Marlborough Cross Test

Table 2.6: Segment 1 (Beacon) Iterative Regression, Selected Results

Rank	Regression Accuracy (%)	Cross Test Accuracy (%)	Regression Parameters
1	97.89	95.80	T_{moving}, $\int\lvert a\rvert dx$, $\int V^2 dt$, $\int V^3 dt$
2	97.87	97.52	T_{moving}, $\int\lvert a\rvert dx$, $\int V dt$
3	97.86	96.47	T_{moving}, $\int\lvert a\rvert dx$, $\int V dt$, $\int V^3 dt$
...	...	...	...
37	96.1	95.22	T_{moving}, $\int\lvert a\rvert dx$
60	89.54	94.08	T_{moving}

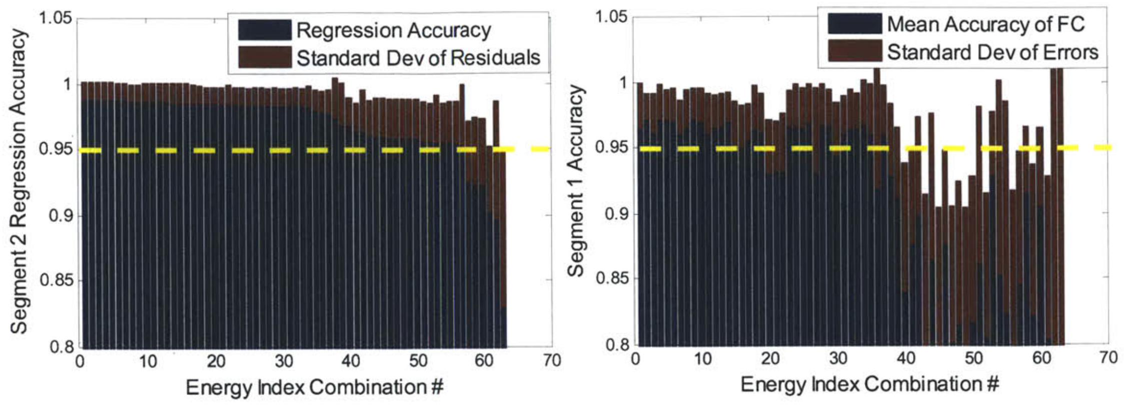


Figure 2.23: Marlborough Street Regression with Beacon Cross Test

Table 2.7: Segment 2 (Marlborough) Iterative Regression, Selected Results

Rank	Regression Accuracy (%)	Cross Test Accuracy (%)	Parameters
1	98.75	96.46	T_{moving}, $\int V dt$, $\int \lvert a \rvert dt$, $\int \lvert a \rvert dx$, $\int V^2 dt$, $\int V^3 dt$
2	98.75	97.18	T_{moving}, $\int V dt$, $\int \lvert a \rvert dt$, $\int \lvert a \rvert dx$, $\int V^3 dt$
3	98.73	96.13	T_{moving}, $\int V dt$, $\int \lvert a \rvert dt$, $\int \lvert a \rvert dx$
...	...	...	...
11	98.63	96.10	T_{moving}, $\int \lvert a \rvert dx$, $\int V dt$
15	98.57	94.98	T_{moving}, $\int \lvert a \rvert dx$
61	90.30	80.39	T_{moving}

The right plots in Figure 2.22 and Figure 2.23 depict the results of using the Energy Index regression coefficients of Beacon Street and Marlborough Street, respectively, to predict the fuel consumption for the parameter data on the opposite street. The order of the parameter combinations is preserved from the plots on the left as a comparison. Overall, the combinations that produced the most accurate regressions also performed well when those regression parameters were tested on the opposite segment. About 50% of the combinations in each test maintained a predictive accuracy of over 95%. This positive result indicates that Energy Index parameters for specific vehicles are street independent, at least for cases in which the segments have comparable road characteristics.

A more detailed breakdown of a few selected parameter combinations is presented in Table 2.6 and Table 2.7. In each table, the 3 best regression results are listed as well as the results for a couple of other preferred combinations, which are groups of parameters that can be easily derived from cell phone accelerometer and map matching data. One unsurprising trend in the results is that models with many parameters tend to have better mean accuracies than those with fewer parameters. This result makes sense in linear regression because parameters with little to no fit for the data will be weighted out with a relatively insignificant coefficient. In other words, the model will not perform any worse by adding parameters, but it will only improve negligibly by adding unnecessary parameters. However, as can be seen in the test plots and the table, more complex models do not necessarily result in a more accurate prediction. Rather, the best performing models vary slightly between the tests. This agrees with the principle of parsimony that is applied to linear regressions, meaning that models should be designed with as few parameters as possible that are able to fully explain a dataset.

The moving time parameter occurs in all of the top ranking models, which is not surprising since it had the strongest correlation with fuel consumption out of all the Energy Indices in Table 2.5. However, the model can clearly be improved by adding additional parameters, since moving time alone does not perform well in all cases when used to predict fuel consumption for new data. Two other parameters that seem to be promising are $\int|a|dx$ and

$\int Vdt$. First, it is important to note that $\int Vdt$ is equivalent to the segment length, which is the same as a constant term in the regression since it should not vary (assuming all regression data comes from one segment). The other term, $\int |a|dx$, is equivalent to the expression $\int |a|Vdt$, which can be easily calculated with accelerometer and sparse GPS data by reconstructing the velocity profile (see Section 2.7). Each of the parameters occurs frequently in the top results. When combined into one model, these three Energy Indices have regression accuracies of over 97% and cross-testing accuracies of over 96%.

So, in determining the finalized Energy Index regression model, we have considered several important criteria. The correlation matrix indicated that most of the parameters were well correlated with vehicle fuel consumption and many of them exhibited high cross correlations as well. Iterative linear regression across all parameter combinations for two road segments yielded an average prediction accuracy of over 95% in most cases. Finally, we have attempted to simplify the model to the least number of components possible while maintaining high regression and prediction accuracies. We selected Energy Indices that both perform well and can be recorded accurately from cell phone accelerometer and GPS data. In Equation 2.11, the proposed finalized model including idling time is presented.

$$\begin{aligned} FC_{seg} &= k_1 T_{idle} + k_2 T_{move} + k_3 \int_{seg} |a| dx + k_4 \int_{seg} V dt \\ &= k_1 T_{idle} + k_2 T_{move} + k_3 \int_{seg} |a| V dt + k_4 L_{seg} \\ FC_{trip} &= \sum_{seg=1}^{s} FC_{seg} \end{aligned}$$

Equation 2.11

where we recognize that $\int_{seg} |a| dx = \int_{seg} |a| V dt$ and $\int_{seg} V dt = L_{seg}$. FC_{seg} is the fuel consumption for one road segment on which the Energy Index values T_{idle}, T_{move}, $\int |a| dx$, and L_{seg} have been calculated and the vehicle-specific regression coefficients k_1, k_2, k_3, and k_4 are applied. FC_{trip} is the total fuel consumption along all of the segments comprising a particular vehicle trip.

2.8.2 Regressing Idling and Non-Idling Components Separately

As stated, the idling time parameter correlates very strongly with idling fuel consumption since the idling fuel rate is nearly constant. So, when including this paramcter in the overall model, there are two options. All of the parameters can either be regressed together, or the idling time and idling fuel consumption values can be filtered and regressed separately. The problem with the first approach is that the idling term is no longer isolated, so its regression coefficient will be tied to the overall fuel consumption instead of the idling fuel consumption specifically. By filtering out the idling time parameter, its regression coefficient is very consistent for individual vehicle models. Table 2.8 lists regression coefficients for the same 2 segments (Beacon Street and Marlborough Street) using both regression approaches. When all 4 Energy Indices are regressed together, the idle time coefficient varies as opposed to essentially remaining constant. However, if the idle time parameter is regressed separately, the idle time coefficients are equivalent while the dynamic parameters seem to vary more. This makes sense since differences between the two data sets can no longer be accounted for by returning unequal idle time coefficients in the regression. However, by regressing idle time separately, the mean prediction error (when the coefficients are used to cross test the other data set) is reduced in both cases and the max prediction error is either slightly reduced or essentially the same.

Table 2.8: Comparison of Combined vs Separate Idling Regression

		Linear Regression Coefficients (x 10^{-5})				Mean Prediction Error	Max Prediction Error
		k_1	k_2	k_3	k_4		
Idling + Moving Regressed Together	Seg 1	12.81	9.8	3.02	1,890	-2.80%	6.02%
	Seg 2	14.54	11.88	2.86	1,630	1.21%	7.67%
Idling + Moving Regressed Separate	Seg 1	13.70	9.84	2.91	1,900	-2.43%	5.67%
	Seg 2	13.70	14.40	3.02	1,000	0.06%	7.85%

2.8.3 Model Performance and Observations

Finally, we apply this model to various datasets in order to compare its relative performance with different vehicles and road types. Table 2.9 lists the regression coefficients for 8 sets of data, including 4 different vehicles and 4 total road segments. As expected, k_1 (coefficient of idling time) is extremely consistent for each vehicle type. The 3 dynamic parameters (k_2, k_3, and k_4) exhibit quite a bit more variation, however. Below each coefficient value in Table 2.9 is the weight of that coefficient in the particular regression. Weight in this instance is defined as:

$$w_{k1} = \left[\frac{(k_1 * T_{idle})}{FC_{tot}}\right] \qquad w_{k3} = \left[\frac{(k_3 * \int |a| dx)}{FC_{tot}}\right]$$

$$w_{k2} = \left[\frac{(k_2 * T_{mov})}{FC_{tot}}\right] \qquad w_{k4} = \left[\frac{(k_4 * L_{seg})}{FC_{tot}}\right] \qquad \textbf{Equation 2.12}$$

$$w_{k1} + w_{k2} + w_{k3} + w_{k4} = 1$$

In other words, the parameter weight describes how much of the total fuel consumption each parameter accounts for on average for that road segment. These weights also vary substantially across vehicle and road segment type. In Figure 2.24, average coefficient weights of the two road segment groups are compared. Idle time is weighted slightly heavier on Memorial Drive, indicating that vehicles idle longer per unit distance travelled. The Beacon-Marlborough route, however, has a larger kinematic metric ($\int |a| dx$) weight, indicating that more energy is spent on average accelerating and decelerating than on Memorial Drive. This conclusion would be consistent with real world observations. The Beacon-Marlborough route has many closely spaced traffic lights for minor intersections as well as somewhat dense traffic, meaning that vehicles are frequently speeding up and slowing down but rarely idling at lights for a long time. Memorial Drive, on the other hand, has several major intersections that frequently force cars to sit and idle for longer stretches. Segment length is weighted approximately the same on both segments, and moving time seems to be weighted heavier on Memorial Drive. Again, this would make sense, since the traffic lights on this road are spaced further apart and it typically has less stop-and-go traffic. Therefore, we would expect velocity profiles to exhibit less variance in speed and for driving time to be weighted more heavily.

Likewise, in Figure 2.25, coefficient weights across vehicle types are compared. Since the Kia Soul was the only vehicle to drive the Beacon-Marlborough route, its distribution is the same as in Figure 2.24. The Toyota Camry, Volvo S40, and BMW 328xi all drove on Memorial Drive. The average coefficient weights of the Toyota Camry and BMW328xi data are remarkably consistent. Each is weighted heavily towards moving time, which would generally be indicative of steady driving in low traffic conditions with a large distance between traffic lights. As mentioned, this is consistent with our observations of Memorial Drive. The Volvo S40 data, on the other hand, weights the kinematic metric most heavily while giving very little weight to driving time. This may occur for several reasons. First, the regression of 3 parameters is being performed on only 6 data points for the Volvo S40, meaning that some of the results may not be as reliable as for the other vehicles. Additionally, all of the data was taken during a single stretch of time in uniformly heavy traffic conditions. The stop-and-go aspect of this traffic would explain the extremely heavy weight on the kinematic metric and idle time.

Table 2.9: Coefficients and Weights for Different Vehicles and Road Types

	Segment Name	Data Points	Linear Regression Coefficients (x 10^{-5})			
			k_1 (T_{idle})	k_2 (T_{move})	k_3 ($\int|a|dx$)	k_4 (L_{seg})
Kia Soul	Beacon Street	19	**13.70** 13.2%	**9.84** 23.8%	**2.91** 39.0%	**1,900** 23.9%
Kia Soul	Marlborough Street	19	**13.70** 22.3%	**14.40** 31.6%	**3.02** 37.6%	**1,000** 8.5%
Toyota Camry	Memorial Drive 1	14	**35.40** 34.3%	**41.50** 42.0%	**1.86** 8.3%	**3,000** 15.2%
Toyota Camry	Memorial Drive 2	14	**36.40** 17.5%	**34.40** 39.3%	**2.76** 18.5%	**3,500** 24.6%
Volvo S40	Memorial Drive 1	6	**16.50** 36.6%	**4.83** 9.1%	**4.88** 33.8%	**2,260** 20.4%
Volvo S40	Memorial Drive 2	6	**16.50** 17.0%	**4.68** 10.5%	**4.37** 45.3%	**2,280** 27.5%
BMW 328xi	Memorial Drive 1	7	**37.60** 28.2%	**47.00** 46.7%	**3.56** 23.3%	**330** 1.8%
BMW 328xi	Memorial Drive 2	7	**37.60** 9.2%	**56.50** 67.0%	**-0.23** -2.1%	**3,200** 25.9%

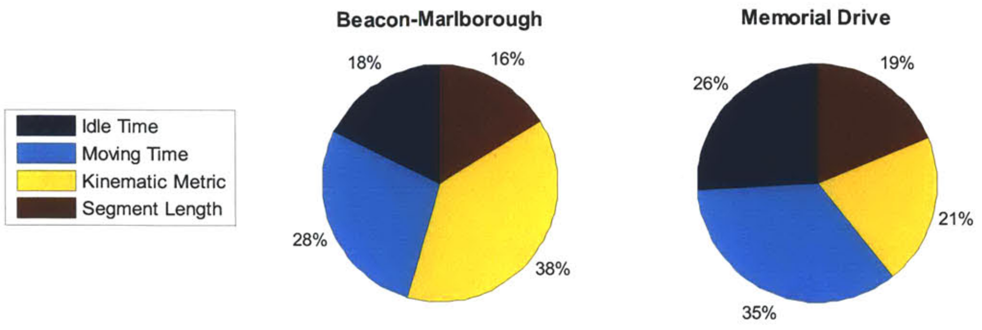


Figure 2.24: Regression Coefficient Relative Weights by Segment

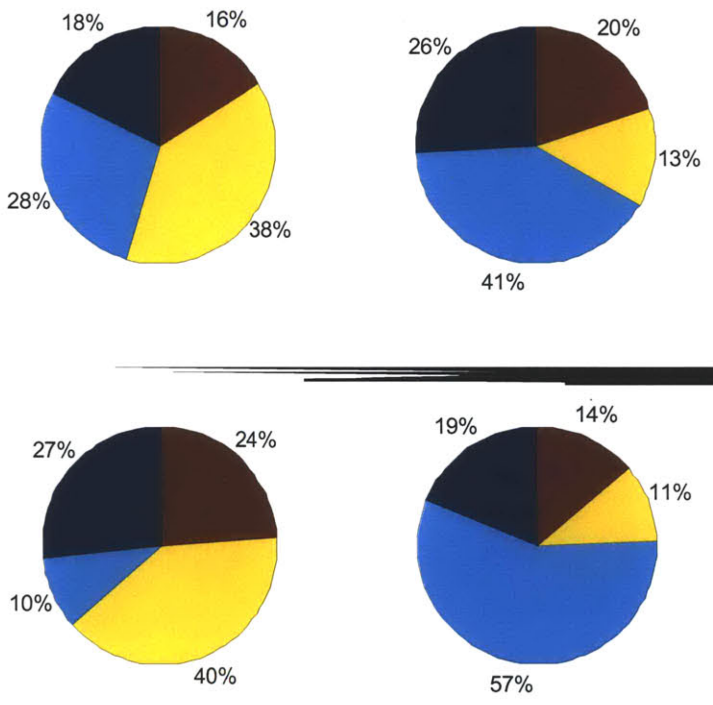


Figure 2.25: Regression Coefficient Relative Weights by Vehicle Type

Table 2.10: Regression and Prediction Accuracies

	Segment	Regression Accuracy	Cross Test Mean Accuracy	Cross Test Mean Error	Cross Test Max Error
Kia Soul	Beacon Street	97.9%	97.5%	-2.43%	5.67%
	Marlborough Street	99.0%	97.2%	0.06%	7.85%
Toyota Camry	Memorial Drive Seg 1	99.1%	97.7%	-1.95%	3.79%
	Memorial Drive Seg 2	98.9%	99.0%	0.21%	3.23%
Volvo S40	Memorial Drive Seg 1	97.9%	94.4%	5.60%	6.76%
	Memorial Drive Seg 2	98.9%	96.6%	-3.44%	7.46%
BMW 328xi	Memorial Drive Seg 1	99.1%	97.2%	0.35%	6.91%
	Memorial Drive Seg 2	99.5%	98.1%	0.69%	3.13%

Finally, we check the estimation error distributions and determine their mean values. If the model is valid, we would expect the errors to exhibit a normal distribution with an approximately zero mean. If this condition holds, significant prediction errors cancel out over the course of several road segments. In Figure 2.26, error distributions for cross tests on the Beacon-Marlborough and Memorial Drive segments are presented. Each distribution is approximately normal, and the mean of the errors is very nearly zero (-1.18% and -0.077%).

2.9 Persistence of Energy Indices

An understanding of Energy Index persistence is vital to the StreetSmart energy consumption model. The importance of persistence becomes especially apparent at the systems level. If an application seeks to determine the most fuel efficient commuting route, for instance, the first step will be to query crowdsourced Energy Index values for the desired road network from the StreetSmart database. These values will be used to estimate fuel consumption using regression coefficients from the vehicle database of the car that most closely matches the one making the commute. In the case of sparse data, accurate persistence statistics will allow the system to fill in the gaps and make the most accurate predictions possible.

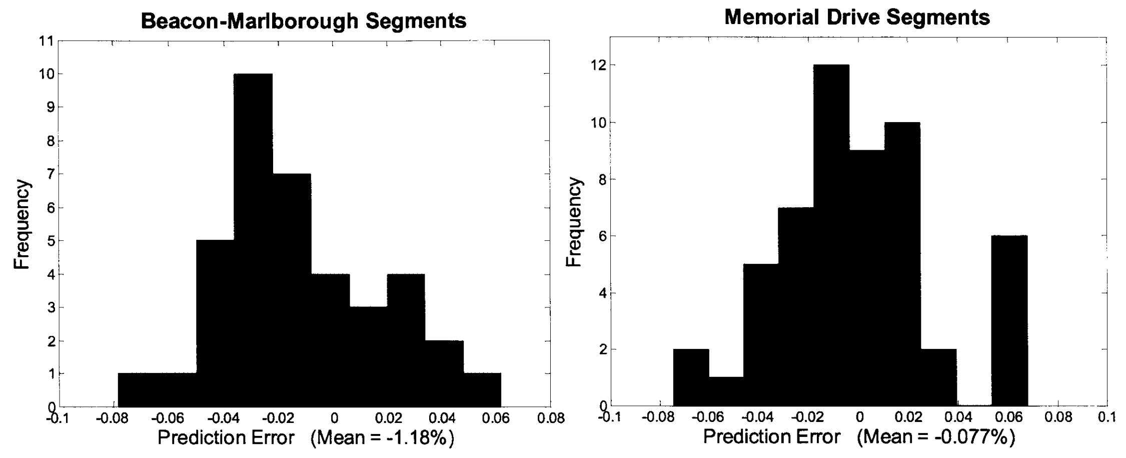


Figure 2.26: Error Distributions of Prediction Cross Tests

There are multiple aspects of persistence that need to be investigated in order to build a robust system. Energy Indices (or traffic conditions) can show persistence in both time and space. For instance, if a traffic scenario occurs on one segment, adjacent segments may be more likely to experience traffic as well, influencing the associated Energy Indices. Some segments adjacent to these segments may also experience related traffic conditions in a diminished form. By examining the Energy Indices of a grid of road segments, it should be possible to identify patterns that indicate spatial persistence of the metrics.

Persistence in time is also critical. Temporal persistence can take two forms: the duration of specific system perturbations as well as the calendar based (day-to-day) repeatability of traffic phenomena. After a perturbation occurs, it is important to understand how long it will affect the Energy Indices and whether the system will return to normal gradually or suddenly. For longer term predictions, several factors can be incorporated. For instance, rush hour traffic is fairly predictable, and it has a measurable effect upon Energy Index values (increased idle time, driving time, and kinematic metric). Major sporting events, holidays, and other occasions can also have predictable effects on certain road segments.

Of course, many traffic events are completely unpredictable, such as a random accident in free flowing traffic conditions that causes a huge vehicle backup. In this case, the system would rely upon a wide enough data collection network to identify the problem relatively quickly and estimate its short term effects. Also, despite not being able to predict all traffic conditions perfectly, the StreetSmart system relies on being right 'on average', allowing noise and other random effects to cancel out or be mitigated over time.

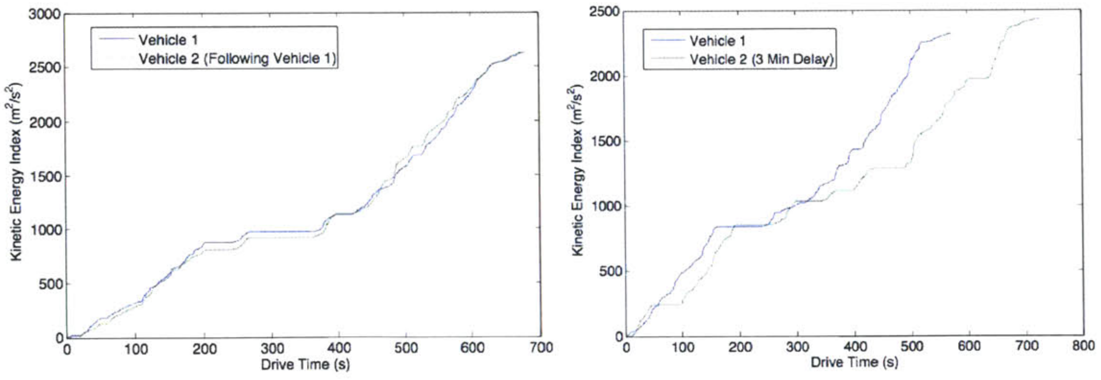


Figure 2.27: Kinematic Metric of Two Vehicles: Lead and Following

In Figure 2.27, the kinematic metric $(\int |a| dx)$ is compared for two vehicles driving through traffic both alongside each other as well as with a 3 minute delay. Each vehicle terminates the drive at the same location. In the first plot, the three moving parameters are all remarkably consistent, which is an excellent sanity check since the vehicles were driving immediately next to each other. Driving time and segment length can also be inferred from these plots. On the right, it is clear that Vehicle 2 catches a couple of extra red lights compared to Vehicle 1 since there are additional flat, non-moving portions of the curve. However, the final kinematic metric values are extremely close, meaning that the traffic conditions persisted quite well during the delay. The only Energy Index term that seemed to vary significantly was idling time, which will have a natural random fluctuation based on second-to-second stoplight signals.

The effect of several different time delay lengths is presented in more detail in Figure 2.28. Here, data is collected on road segments for two vehicles that start from the same location with 0 to 3 minutes of delay in between. Both the kinematic metric and the moving time vary only a few percentage points on average, with a max variation of 20% for the kinematic metric after a 2 minute delay.

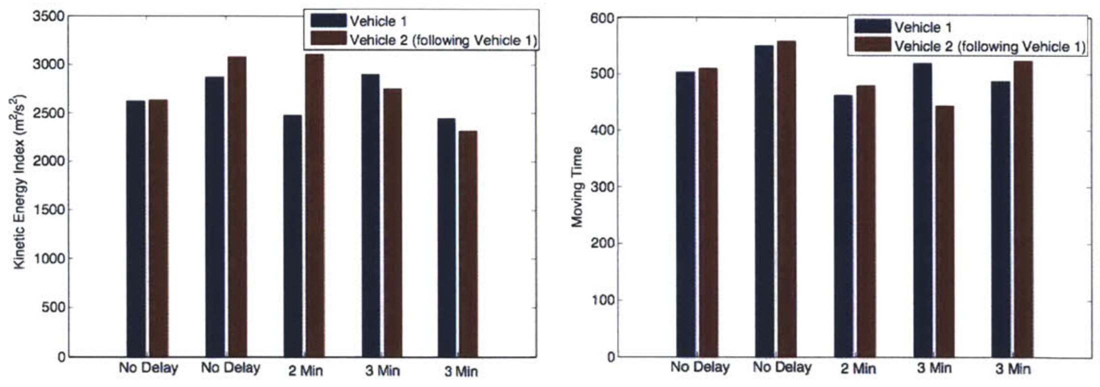


Figure 2.28: Time Persistence of Kinematic Metric with Various Delays

Chapter 3 StreetSmart Participatory Sensing System Overview

In isolation, the regression model we developed in Chapter 2 is very useful for vehicle energy estimation applications. We can use the Energy Indices in combination with the stored vehicle regression coefficients to predict a vehicle's fuel consumption in heavy traffic conditions with a high level of accuracy. We achieved this by using real world measures of traffic to improve the traditional physics-based models of fuel consumption. This approach allowed us to split street segments into discrete chunks without introducing dummy variables for traffic lights and stop signs as some models in the literature have done.

Additionally, we optimized the model specifically for smartphone accelerometer and GPS sensor data. Therefore, the 4 Energy Indices we selected will be valuable at the systems level in which many distributed smartphones are contributing data to the project. In this chapter, we will describe the StreetSmart System concept in the context of participatory sensing networks. Section 3.1 provides background on participatory sensing networks, the motivations of potential users, and reviews some projects that have previously been attempted in the literature and industry. Section 3.2 addresses the overall objectives of the StreetSmart system from a high level view. Finally, Section 3.3 describes our vision of the StreeSmart system architecture in detail.

3.1 Participatory Sensing

The concept of participatory sensing was introduced in section 2.1. Research projects such as GreenGPS, CarTel, and the Mobile Century [7,6,8] have been pioneered at universities to collect traffic data and for estimating the fuel consumption of vehicles based on average speed measures. Many other participatory sensing networks have been implemented at universities with varying objectives. For instance, in [14], the authors develop Micro-Blog, which is described as a tool that could be used for sharing and browsing global data. The authors of [40] develop ParkNet, which employs a network of vehicles with passenger side ultrasonic rangefinders to map parking spot occupancy.

A participatory sensing system relies on a dual purpose user base. First, a potentially small but well distributed portion of the users must be willing to collect and volunteer their own data to create an information network of some type. The entire user base (which may be significantly larger than the number of users who volunteer data) can then make use of this data for various applications. The application becomes increasingly useful as enough data contributors begin to participate to reach some minimum threshold.

What would motivate some users to volunteer their data while others do not? There are many possible answers to this question, and participatory sensing projects have attempted a range of strategies. For instance, users can be paid money proportional to the volume or quality of the data that is provided. This is the method that some university research projects employ to get an adequately sized user network. It also helps in circumventing the chicken and the egg problem of whether data or users come first to a successful application. Later on, users can often be motivated to provide data if they are granted free or discounted access to a useful application.

In the case of some already widely distributed applications, an opt-in or opt-out participatory sensing project can be easily implemented. If it presents extremely low barriers of entry to the end user as well as no apparent downside, users may be willing to supply data. In a smartphone application, it is also possible to run certain data collection and uploading tasks in the background with no active involvement by the consumer. This method may only be compromised if serious concerns over battery life or privacy arise.

The smartphone is well suited to serve as a participatory sensing platform, as the embedded sensors in these phones are rapidly improving over time in both quality and scope. Additionally, these devices are becoming ubiquitous worldwide, so any smartphone-based participatory sensing network has the potential for extremely wide distribution. In the context of vehicle data, a smartphone can capture detailed acceleration profiles and GPS locations of vehicles along a road. Other dedicated GPS and OBD-II logging devices, such as those sold by companies like Garmin and Auterra, can record vehicle data at a much higher resolution. However, the distribution of dedicated OBD-II loggers will never come close to that of smartphones unless some government mandate emerges requiring them. Users and OEM automotive manufacturers alike have shown little interest to incur the extra expense it would take to provide dedicated logging units in new vehicles, and privacy concerns would likely prevent researchers from gaining API access to user data without strict controls in place.

3.2 StreetSmart System Objectives

We have already developed a model to act as the foundation of the StreetSmart system. However, this is only the base component of the overall participatory sensing system. In this section, we outline the central objectives of StreetSmart and clarify its core functionalities before elaborating on the details in subsequent sections. The elements of the StreetSmart system beyond the central model are not completely implemented in this thesis. The full development and launch of StreetSmart is left as future work, and this chapter is meant to propose a framework for this development.

Table 3.1: High-Level StreetSmart System Objectives
1) Collect and process OBD-II data from a wide cross section of vehicle models and create a vehicle database of regression coefficients for the 4 Energy Indices versus fuel consumption.
2) Crowdsource smartphone accelerometer and GPS data from moving vehicles and store the 4 Energy Index values in a real-time street segment database.
3) Release an API providing access to the vehicle and Energy Index databases for applications such as traffic management, electric vehicle range estimation, hybrid vehicle drivetrain management, and others.

In Table 3.1, the high level objectives of StreetSmart are listed. First, the system aims to construct a wide ranging database of vehicle-specific regression coefficients. This data must be collected from OBD-II and GPS loggers installed in vehicles, since a measure of actual fuel consumption is needed to perform the regression. Volunteers, partner fleets, or paid individuals could supply the data. Regression coefficients will be calculated and stored for as many vehicle and street types as possible. In Section 2.9, we demonstrated that Energy Index regression coefficients were persistent in time for similar road types. So, we must store regression coefficients for each vehicle model as well as for different street types across the same vehicle model. We can be confident, though, that the regression coefficients remain persistent for a particular vehicle and street type combination.

Second, the system must collect a large amount of driving data from the general public in order to create real-time snapshots of traffic conditions. A study by BMW estimates that a 4-10% penetration rate would be required to sufficiently characterize real-time traffic data in a useful manner [41]. To capture major incident information on highways, a 4% adoption rate would be adequate. Monitoring urban traffic conditions, though, requires an adoption rate closer to 10%. Fortunately, any moving vehicle carrying a smartphone is capable of providing useful data. Eventually, the StreetSmart crowdsourcing tool could be distributed as either a standalone application run by users or as a package with other traffic related applications. The most critical feature for the purposes of data collection is to have a background thread running that can estimate whether or not a smartphone is currently in motion and likely to be inside a vehicle. If that condition is met, the smartphone would communicate the 4 Energy Indices back to the server in real time over the cellular network.

Finally, the data collected for StreetSmart should be used in conjunction with applications developed by both the StreetSmart researchers as well as outside developers through an external API. By accessing the vehicles database and the Energy Index database, developers could make predictions about the energy expenditures of future trips or implement a variety of other applications. The Energy Indices themselves could be used to report real time traffic conditions on specific urban road networks for which only sparse data is currently available. The vehicles database could be employed to compare the energy consumption of different vehicle types on the same road network at different times of day, predicting which vehicles may be better to use in rush hour conditions versus normal conditions and noting any discrepancies. In Section 3.3, we go into more specific detail about the system architecture of StreetSmart, including data handling and processing methods.

3.3 System Architecture

With a few key exceptions, StreetSmart has a system architecture similar to many participatory sensing applications. For instance, we must handle two incoming streams of data. The smaller, OBD-II data stream must be processed and regressed periodically for new vehicles in order to maintain up-to-date Energy Index coefficient values. The much larger stream of smartphone accelerometer and GPS data must be collected over the cellular network, processed to calculate up-to-date Energy Index values, and stored in a highly scalable manner. There must also be a strict emphasis on maintaining user privacy since the datasets are potentially invasive.

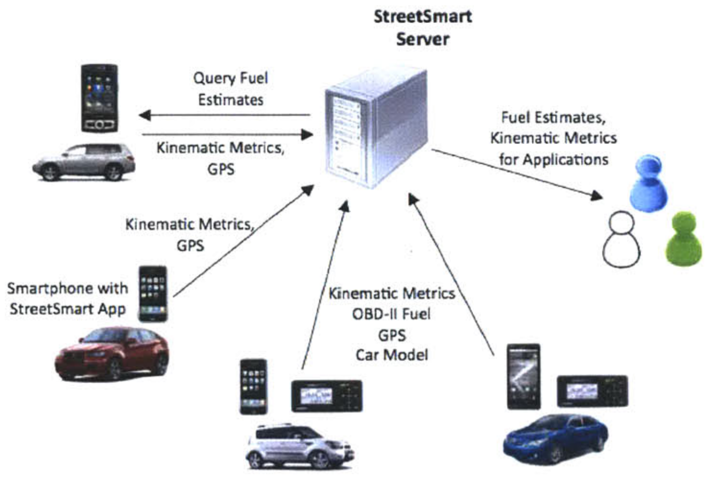


Figure 3.1: StreetSmart System Architecture

In Figure 3.1, a visualization of the data flows within the StreetSmart system is provided. The StreetSmart server contains a collection of databases and processing algorithms to interface between the various data streams. Some vehicles will provide the Energy Indices from smartphone sensors while others contribute OBD-II fuel and velocity data. On the applications side, some users will directly access the StreetSmart system to check traffic conditions as reported by the Energy Indices. Other users could access applications that make use of the StreetSmart API to implement novel traffic and energy solutions.

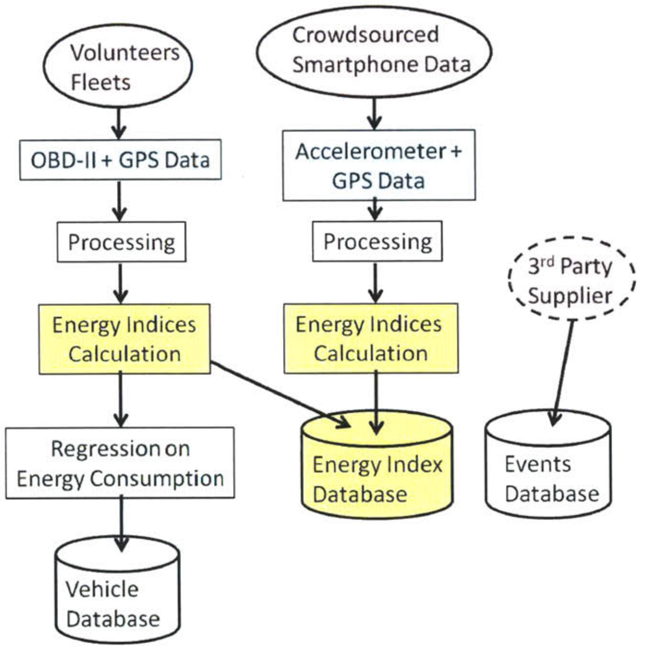


Figure 3.2: Data Flow Within StreetSmart

A more detailed breakdown of the data collection steps is provided in Figure 3.2. Data sources are highlighted in red, values associated with the Energy Index database are highlighted in yellow, and values associated with the vehicle database are highlighted in grey. Volunteers and partner fleets provide the OBD-II data for calculating the regression coefficients in the vehicle database. The associated Energy Indices of the OBD-II data, though, are also stored in the Energy Index (EI) database. The 'processing' step present in the first two columns was outlined in detail in section 2.6 (specifically laid out in Table 2.4). StreetSmart may also contain an events database that enables it to intelligently adjust predicted Energy Indices for specific dates and times. Known events such as sports games, road maintenance, large accidents, and others could be kept in this database for use in some applications.

3.3.1 Client-Side vs. Server-Side Processing

We must decide which of the processing steps will occur on the client side (participant smartphones) and which will occur on the StreetSmart server side. On one hand, it is important to limit participants' data transmission costs, which implies that processing should be done locally on smartphones. On the other hand, there are multiple steps required to calculate the Energy Indices, and fewer system resources may be required to simply transmit the raw data over the cellular network.

Local processing would initially seem to limit required network transmission parameters to the unique road segment ID's and the 4 Energy Index values of each road segment. In reality, there are added transmission costs with the local processing approach due to the requirements of reverse geocoding. Every GPS data point must be transmitted over the network to the reverse geocoding database, at which point it would be processed and returned to the phone with the street and intersection data. Reverse geocoding is necessary in order to split a driving route into discrete road segments according to the database. So, in attempting to save on data transmission costs through local number crunching, there is additional transmission overhead for the GPS data. However, since the accelerometer data is recorded at a much higher sampling rate than GPS, there would still be a net overall savings on data transmission. In terms of processor costs, though, local computation leads to a major drain on resource availability as well as battery life. For StreetSmart, this challenge is especially relevant since the participatory sensing application will likely be running as a background application and may be afforded a small processing and power budget.

Conversely, remote processing would transmit raw GPS and accelerometer data to the StreetSmart server as it is recorded. Assuming the sampling rate of the accelerometer is significantly higher than the GPS, the transmission costs of this approach would outweigh the first approach. However, the processing burden would be offloaded from the smartphone to the StreetSmart server. One advantage of this approach would be the higher confidence level in our stored data, which would contain raw sensor measurements instead of processed numbers that may contain hidden errors from reverse geocoding or other sources.

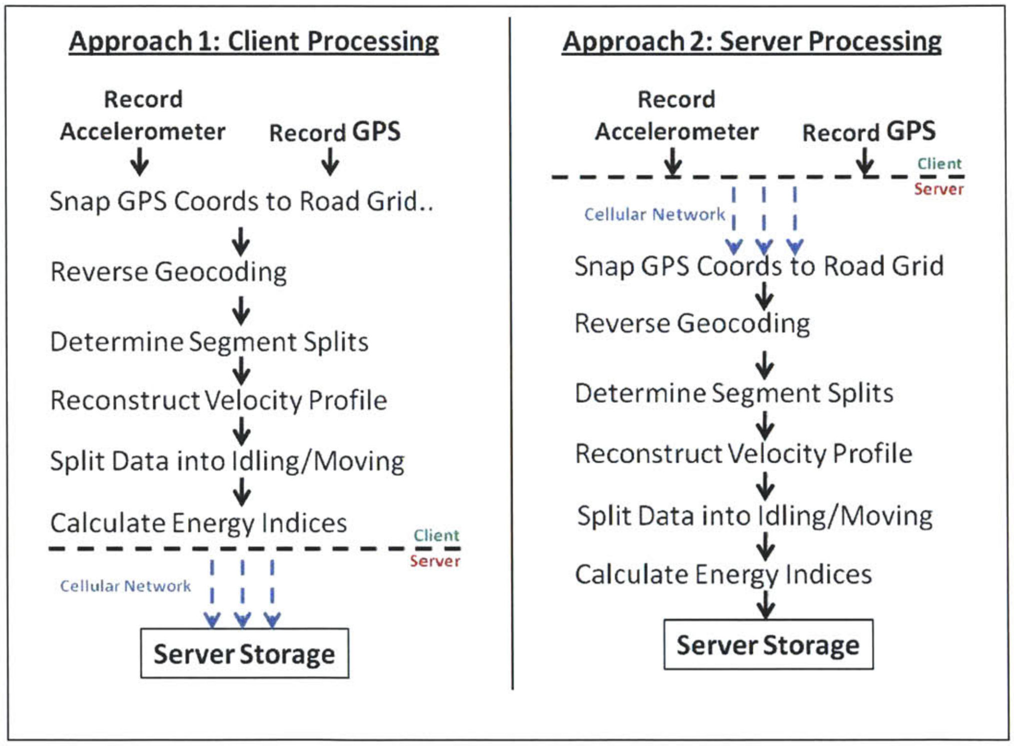


Figure 3.3: Client-Side Processing vs. Server-Side Processing of Smartphone Data

Figure 3.3 compares these client-side and server-side approaches. In each case, the smartphone records accelerometer and GPS data at the necessary sample rates. In the first approach, the data processing steps are carried out locally on the phone before the data is transmitted to the StreetSmart server. The second approach illustrates the transferring of raw data to the server, where the data processing steps occur. However, these two absolute approaches are not the only options. We can use a hybrid of the client/server side strategies along with some preprocessing to minimize both transaction costs as well as phone processor usage. Figure 3.4 illustrates the data flow of this type of an approach.

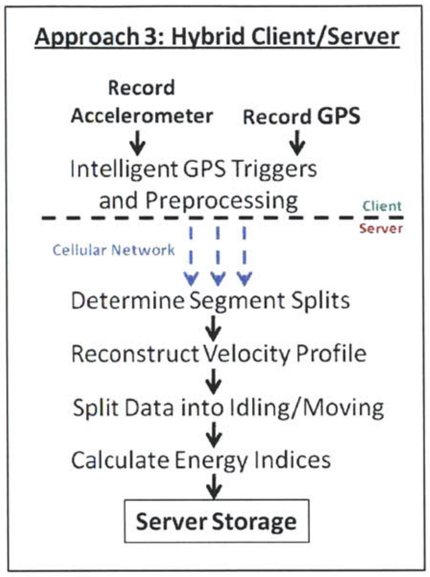

Approach 3: Hybrid Client/Server
Record Accelerometer
Record GPS
Intelligent GPS Triggers and Preprocessing
Client
Server
Cellular Network
Determine Segment Splits
Reconstruct Velocity Profile
Split Data into Idling/Moving
Calculate Energy Indices
Server Storage

Figure 3.4: Hybrid Client/Server Processing

Instead of taking GPS measurements at an even sampling rate and then transmitting all of the raw data to the server for reverse geocoding, GPS would be sampled intelligently. For instance, a vehicle can predict when it is approaching the next intersection node using dead reckoning accelerometer techniques. It would only record GPS samples as it anticipates approaching a node, saving substantially on GPS energy costs. After a street node overlap is detected with the GPS signal, the accelerometer data between the last two nodes would be transmitted to the server along with the segment ID. Subsequently, a list of distances to all adjacent nodes would be transmitted back to the smartphone, which would use dead reckoning to predict when another node is being approached.

Using this strategy, GPS sampling would be minimized, boosting the application's energy efficiency. The smartphone would also carry out very little processing besides simple dead reckoning techniques, limiting the drain on processing resources. Table 3.2 summarizes our above comparisons of the 3 data management approaches in terms of local processing cost, transmission cost, and hidden advantages or disadvantages. The hybrid client/server strategy compromises on both processing and transmission loads imposed on the user's smartphone, but it seems to offer the most balanced solution in terms of energy management and local resource usage.

Table 3.2: Comparison of 3 Data Processing and Transmission Approaches

Approach	**Local Processing Cost**	**Transmission Cost**	**Hidden Advantage or Disadvantage**
Client Processing	High	Low-Medium	Data Transmission for reverse geocoding and map matching
Server Processing	Low	High	High GPS Energy Cost
Hybrid Client/Server	Medium	Medium	Smart GPS Usage, Server Assistance

3.3.2 Application Queries

The StreetSmart server will store several databases that should be accessible to queries through user applications. The server must utilize matching algorithms to process a user query that requests the predicted energy consumption of a vehicle model for a specified route at some point in the future. In Figure 3.5, the structure of a typical query is presented. The user provides the vehicle type, road segment, and desired time. The server calls the findBestEI() routine to identify and transmit the system's best guess for the Energy Index values on that road at the desired time. Additionally, the findBestVehicle() routine is called in order to find the regression coefficients that will provide the best possible results for a given vehicle and road type.

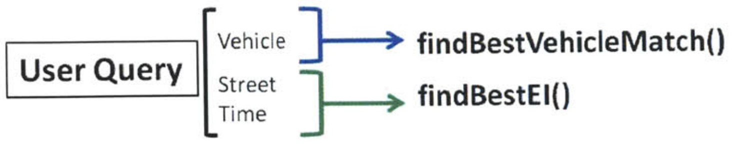


Figure 3.5: StreetSmart Application Query

In Figure 3.6, the findBestEI() algorithm is laid out in more detail. The routine runs 2 queries on the Energy Index (EI) Database. The first query loads all EI data points that match the road segment in the query, sorting them by relevance. In this case, data points closer in time to the queried time value are ranked as more relevant. Interestingly, a time-of-day match from a different calendar date may be more relevant for an EI prediction than data points from the same calendar day but further away on the clock (i.e. rush hour traffic usually occurs during the same 2 hour afternoon window). After the algorithm has identified the most relevant same-street data, it looks for data on surrounding road segments that may rank higher in relevancy in case there is a lack of data during the desired time on the same street. The algorithm sorts the EI's by relevance and returns $EI_{estimate}[]$, the 4 Energy Indices with the highest confidence level.

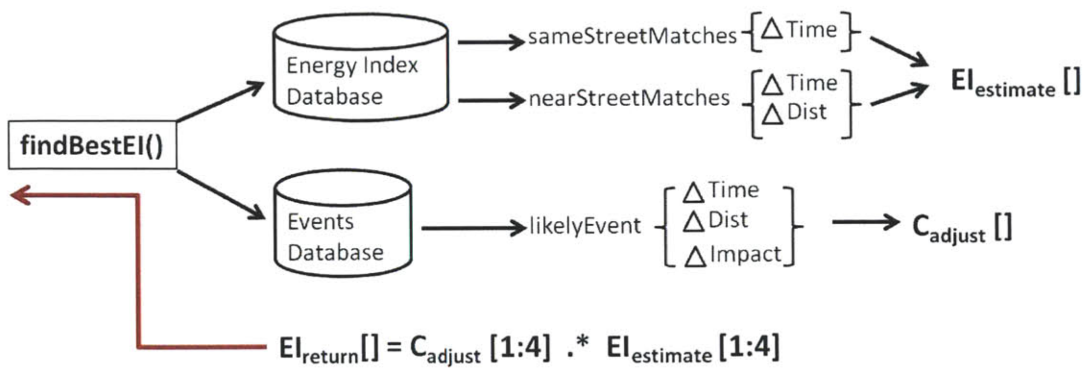


Figure 3.6: findBestEI() Algorithm

In parallel, the events database is searched for any known events that may affect the Energy Index values in a predictable manner. For instance, a scheduled sporting event or known holiday may cause a shift in traffic conditions that results in more stop-and-go driving on certain arterial road segments. A likely event is weighted by its relationship to the query in terms of time, distance, and expected impact. By taking each of these factors into account, the function assigns a value C_{adjust} to the event as a predicted scaling factor for each of the EI's. The routine returns an array EI_{return} [] that is the piecewise multiplication product of C_{adjust}[] and $EI_{estimate}$[].

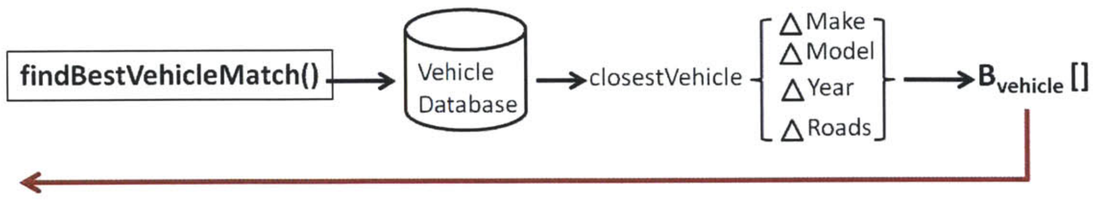


Figure 3.7: findBestVehicleMatch() Algorithm

It is likely that more vehicle makes and models will exist in the world than the number of volunteers or fleets collecting OBD-II data for the StreetSmart system, leading to a sparse data condition with our regressions. Therefore, it will be necessary to apply a matching algorithm to find the best vehicle regression coefficients to use when estimating energy consumption. The algorithm findBestVehicleMatch() is illustrated in Figure 3.7. The vehicle database is queried and the regression coefficients of the most relevant vehicle are returned, ranked on differences in make, model, year, and road type.

Table 3.3: Vehicle Coefficient Selection Hierarchies for Sparse Data

Make	Make	Make
↳Model	↳Model	↳Road Type
↳ Year	↳Road Type	↳Model
↳Road Type	↳ Year	↳ Year

The exact method of determining the relevancy of specific coefficient sets is dependent on several factors. First, it is important to note, as we found in Section 2.9, that regression coefficients for the same vehicle can vary for significantly different road types. Conversely, on roads that have roughly the same density and distribution of traffic lights, the coefficients do not vary significantly. So, when an exact vehicle and road type match cannot be found in the vehicle database, we must select coefficients from the layers of data that provide the highest available regression accuracies. We would expect the regressions to become more reliable as the regression dataset is derived from more specific vehicle type matches. For instance, the coefficients obtained by regressing the fuel data of a group of 2007 Ford F-150 trucks would almost certainly be more reliable for predicting the fuel consumption of other 2007 Ford F-150's than coefficients obtained by regressing the fuel data of trucks in general.

Road type, however, carries an unknown level of relative importance in this parameter hierarchy. For instance, a dataset of Ford F-150's on only one road type but with a mix of various model years might have a higher regression accuracy than a dataset of 2007 Ford F-150's on multiple road types. Table 3.3 presents 3 different approaches to funneling vehicle parameters in the case of sparse data. In the second and third cases, road type is simply moved up in importance by one level each time. The vehicle database can be set up in such a way that when the findBestVehicleMatch() algorithm is executed, all sparse data combinations are considered in order to find the most accurate regression coefficients possible for a specific vehicle request. Figure 3.8 illustrates the coefficient selection matrix for the first parameter hierarchy in Table 3.3. A vehicle query iterates to the level of its most specific matching coefficients. For instance, if a 2005 Toyota Camry was being queried, the coefficients for $B_{Comp\text{-}Camry}$ [] would be returned in this case because no regression data is available for the 2007 model year. If, though, we queried 'Road Type 1' and this data was available, the regression accuracy of the second column of Table 3.3 may provide better results. The findBestVehicleMatch() algorithm compares the regression accuracies across each of these different approaches for any data query and provides the best estimation for EI coefficients.

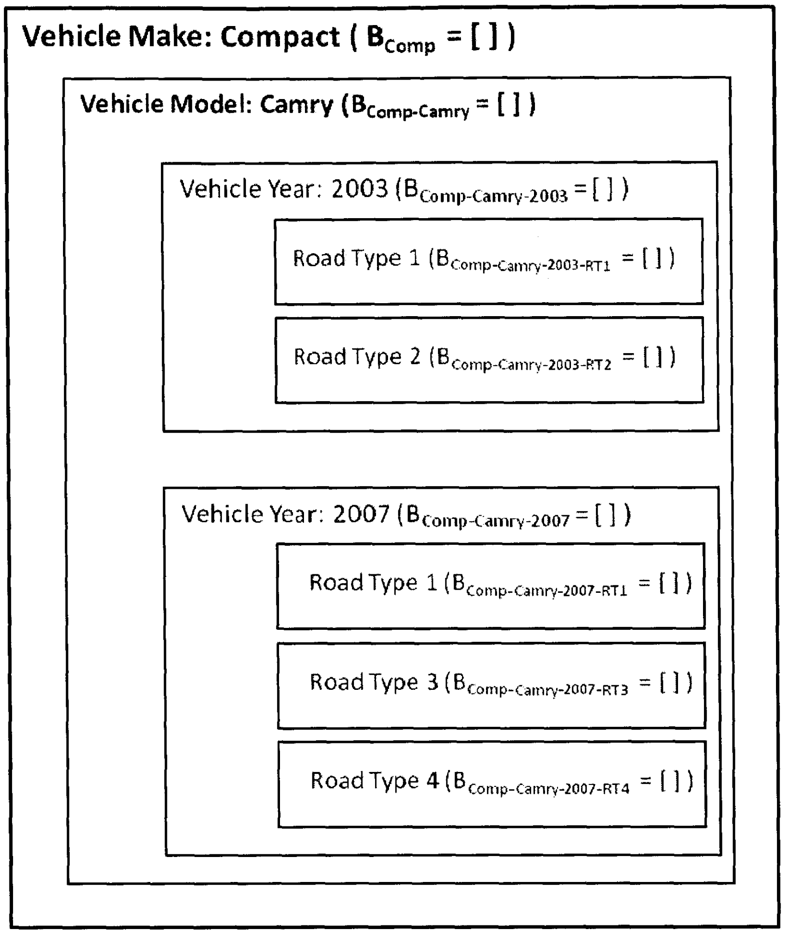

Vehicle Make: Compact (B_{Comp} = [ ])
Vehicle Model: Camry ($B_{Comp-Camry}$ = [ ])
Vehicle Year: 2003 ($B_{Comp-Camry-2003}$ = [ ])
Road Type 1 ($B_{Comp-Camry-2003-RT1}$ = [ ])
Road Type 2 ($B_{Comp-Camry-2003-RT2}$ = [ ])
Vehicle Year: 2007 ($B_{Comp-Camry-2007}$ = [ ])
Road Type 1 ($B_{Comp-Camry-2007-RT1}$ = [ ])
Road Type 3 ($B_{Comp-Camry-2007-RT3}$ = [ ])
Road Type 4 ($B_{Comp-Camry-2007-RT4}$ = [ ])

Figure 3.8: Example Vehicle Coefficient Selection Hierarchy

Finally, after the predicted EI's and best-fit regression coefficients are selected, the predicted vehicle energy for the queried road segment can be calculated by multiplying and adding these values together as shown in Equation 3.1. Of course, the user may also simply query the EI values directly for use with traffic-related applications which are not specifically interested in vehicle energy consumption.

$$\text{VehicleEnergy}_{predict} = B_{vehicle}[1:4] * EI_{estimate}[1:4] \qquad \textbf{Equation 3.1}$$

3.3.3 Smartphone Vehicle Detection

It would be impractical to expect StreetSmart users to initiate and exit the data collection application consistently when operating their vehicles. There must be nearly zero effort demanded of end users in order to successfully collect a large amount of data with this method. The best implementation, then, will likely be an application that uses a background process for vehicle motion detection. This algorithm will have to actively poll accelerometer data while analyzing it for the signature motion profile of a vehicle. It will have to filter out other types of motion, such as walking, riding on a train, or waving the phone in the air. While the full implementation of this method is left as future work, we have conducted initial analysis leading us to believe that such detection methods are possible using real-time smartphone data.

Chapter 4 Future Applications and Conclusions

In this thesis, we have developed a novel energy consumption model for road vehicles in the context of a participatory sensing network named StreetSmart. This system crowdsources vehicle OBD-II data and smartphone sensor data that is collected while participants are driving. In order to formulate the model and the broader system architecture, we collected a significant amount of OBD-II driving data around the city of Boston. In Chapter 2, a regression model for the fuel consumption of internal combustion engine vehicles was developed and presented in great detail. In Chapter 3, the framework of the StreetSmart system was explored thoroughly while leaving the full implementation for future work. In this chapter, we present additional applications that might be developed for StreetSmart as well as some concluding thoughts. In Section 4.1, we introduce recent trends in vehicle hybridization and electrification, and we discuss possible integration opportunities for the StreetSmart system with these modern drivetrain technologies. Section 4.2 discusses other future applications of StreetSmart, including congesting taxes and traffic data reporting. Section 4.3 outlines challenges that we expect in the implementation of StreetSmart, and Section 4.4 briefly reviews our steps for future work.

4.1 Hybridization and Electrification

So far, we have only considered applications related to pure internal combustion engine (ICE) vehicles. However, many would argue that the future of transportation lies with hybrid electric (HEV), plug-in hybrid electric (PHEV), and full battery electric vehicles (EV) due to the vast increase in powertrain efficiency achieved with these compared to ICE vehicles. Hybrids have become significantly more mainstream since the release of the original Toyota Prius in

1997. An HEV is able to take advantage of the high energy conversion efficiency of the electric motor while also using the incredible energy density of gasoline to provide a normal vehicle range. This combination eliminates the "range anxiety" that many people feel with pure battery electric vehicles, which are limited in most cases to between 100-150 miles of range because of the high cost and low energy density of modern lithium-based batteries.

4.1.1 Plug-in Hybrid Electric Vehicle (PHEV) Drivetrain Optimization

Normal HEV's use a control strategy known as "charge sustaining", which maintains a set target for the battery state-of-charge (SOC). The SOC is allowed to vary in order to keep the ICE operating at the most efficient point possible while also meeting the power demands of the vehicle. Because of the limited battery capacity, though, the maximum energy fluctuation of the pack may only be 1-3 kWh. Of course, this varies based on the size of the battery back as well as the hybrid architecture in place (mild, series, parallel, or series-parallel). However, it is important to note that the batteries in HEV's can only be recharged through two means: regenerative braking and the ICE turning a generator.

PHEV's, on the other hand, introduce an additional degree of freedom with respect to energy storage because they are able to "plug-in" to recharge the battery pack. This ability is necessary in order to support the much higher battery capacity that is available (typically an order of magnitude difference – 10-15 kWh instead of 1-3 kWh). The simplest means to implement a PHEV control strategy is to first operate in "charge depletion" mode. The electric motor carries the full burden of the drivetrain except when the ICE is needed to provide peaking torque. In a series PHEV architecture, this would be known as "all electric" mode, which is why these cars are oftentimes called range extended electric vehicles (REEV). Parallel or series-parallel PHEV's can operate in "blended mode" during charge depletion, only allowing the engine to turn on and assist in its most efficient operating range [42]. After the SOC lowers to some threshold value, the control strategy switches to "charge sustaining" mode and operates similar to the regular HEV.

A mature version of the StreetSmart system could integrate well with an advanced PHEV control strategy that is currently being explored in the literature. Some research efforts have simulated "globally optimal" control strategies for parallel architecture PHEV's that blend the "charge depletion" and "charge sustaining" modes using dynamic programming and other methods [43,44]. This type of control strategy makes three assumptions that must hold to improve efficiency: the vehicle trip in question must be at least long enough to deplete the full battery in all-electric mode, a charging outlet must be available at the destination, and we must have access to street and traffic information about the intended route. If this is the case, and a priori information about the route is available through a system like StreetSmart, a dynamic programming algorithm can be employed to determine the optimal drivetrain management strategy.

Since the electric motor always converts energy more efficiently than the internal combustion engine, it may seem intuitive that the optimal control strategy would simply be able to run the vehicle in all-electric mode at first followed by charge sustaining mode later, as described previously. However, since the internal combustion engine attains peak efficiency for very narrow RPM ranges, restricting engine operation only to these regions can lower the overall energy consumption of the vehicle. So, the optimal battery discharge profile will most likely appear relatively steady until the end of the commute as the low SOC point is reached.

As one example in which the optimal control strategy would differ from the first strategy, consider the case of a 100 mile commute that consists of a highway section followed by a city section. The engine would perform poorly in the city (stop-and-go, low speed driving) but would be forced to operate at a high duty cycle there if the all electric mode was exhausted on the highway. Using the optimal control strategy, an adequate portion of the battery SOC will be reserved for the city driving that will optimize the overall efficiency of the ICE operation for the entire trip. StreetSmart could come into play by providing the traffic data needed to perform real-time dynamic programming for optimizing SOC management during commutes. Drivers would select the final destination and indicate whether or not charging would be available afterward. If so, the system would optimize the SOC discharge profile based on the Energy Indices being reported along the predicted route.

Of course, there are many practical issues that must be considered before such a drivetrain control approach could be implemented and integrated with a system like StreetSmart. First, and most importantly, PHEV's would have to be sold by car manufacturers and move into the mainstream. This is certainly likely within the next 5 years, as several manufacturers plan to transition popular HEV models to have plug-in capability. The technology must also mature using ordinary control techniques before any advanced, route-aware strategies are actually implemented. Finally, integration with a system like StreetSmart would require a thorough investigation into the efficacy and persistence of Energy Index values, since the dynamic programming algorithms would rely on their predictability. Additional work would have to be done analyzing the inherent uncertainty of traffic microstructure and how this impacts the usefulness of the real-time drivetrain optimization. For instance, if we use dynamic programming to design a control strategy around unreliable data, there would be some breakeven point of data quality for which the "optimal" method performs the same or worse than the default method.

4.1.2 Battery Electric Vehicle (BEV) Range Estimation

Although PHEV's do offer an excellent transition away from pure ICE vehicles, the holy grail of transportation efficiency is making the leap to pure BEV's. These vehicles take full advantage of electric powertrain efficiency and performance, delivering instant torque from rest and operating in near silence. However, the extreme decrease in the energy density and increase in cost of modern batteries compared to gasoline is a major drawback. The first generation of production electric vehicles, which have just begun to reach the market over the last couple of years, typically feature a maximum range of about 100 miles. The exception is the Tesla Roadster, which drives 230 miles in one charge but sells for over $109,000. The lack of available energy storage capacity for affordable BEV's in combination with battery recharge times of many hours has led to the phenomenon known as "range anxiety." This is a condition that supposedly will prevent many consumers from taking a chance on BEV's because of their fears of being stuck in a location for hours after running out of charge.

As demonstrated in Section 2.3, the energy consumption of a vehicle varies dramatically based on real-time traffic conditions. Aerodynamic drag force, inertial resistance to acceleration, and powertrain efficiency will all change based on the exact velocity profile that is followed over a road segment. So, it is important to note that even if a BEV has a specific range rating, the exact distance that it is able to drive on a single charge will be determined by the driver and the roads that are taken. This uncertainty is a contributing factor to range anxiety, especially since the "remaining range" estimation algorithms in electric vehicles are not currently calculated by taking any real-time factors into account.

StreetSmart could improve dynamic range estimation of BEV's by using real-time Energy Index estimates combined with transfer functions like in Chapter 2. Drivers would select an intended destination or route and StreetSmart would calculate the predicted energy usage over the entire path. The estimate would be far more accurate than a static estimate of range based on some arbitrary factor such as average energy use over the previous 50 miles. Furthermore, integration with StreetSmart would allow for vehicle makers to develop rich user interfaces for in-vehicle displays. These UI's could do things like illustrate the predicted energy use on upcoming road segments that are visible on the navigation map. Drivers could also have more peace of mind from a reliable algorithm (combined with actual real-time data) that would tell them if the available charge is adequate to reach home or the next charging station.

4.2 Additional StreetSmart Applications

The StreetSmart System was designed with a broad array of potential applications in mind. The first and most obvious application, in line with the model developed in Chapter 2, is the prediction of fuel consumption values for ICE vehicles in real-time. These estimations can be used to calculate the "greenest path" for an arbitrary vehicle at any time of day. However, in addition to fuel estimation as well as the PHEV and BEV applications, there are several other traffic related possibilities for which to apply the crowdsourced data.

4.2.1 Traffic Reporting and Estimation

A road segment's Energy Indices provide insights about traffic in a multidimensional context. As discussed in Section 2.3, average speed by itself is not an adequate characterization of traffic conditions when the goal is to understand the full relationship between traffic and energy consumption. The Energy Indices do contain a measure of average speed (moving time and segment length). Additionally, the kinematic metric $\int |a| dx$, which is proportional to the inertial energy expended by accelerating and decelerating the vehicle, represents an intuitive method of characterizing the fluctuating motions of traffic. The separation of idle time and moving time, furthermore, provides a much better quantitative insight into the nature of traffic than just an average speed, which blends moving and idling components together.

Traffic data providers could use StreetSmart Energy Indices to improve their reporting and distribution of real-time traffic information, especially on arterial roads and urban streets that typically are difficult to monitor or predict. Currently, most traffic is simply reported to consumers in 2 or 3 tiers: heavy (red), clearing (yellow), or free (green). Energy Indices could enable traffic providers to supply greater detail such as the estimated impact of specific traffic conditions on fuel consumption or a precise idea of how the condition will clear over time. As a more complete and accurate measure of vehicle traffic, StreetSmart could be a potentially valuable tool for fleet operators, delivery services, and other logistics providers that rely on up-to-the minute forecasts of road conditions and their impact on the bottom line.

4.2.2 Congestion Pricing

Simple forms of congestion pricing based on time-of-day traffic predictions have been implemented in European cities such as London. Some cities in the United States like San Francisco increase tolls on bridges during known times of high traffic. This price discrimination, while crudely effective at mitigating rush hour traffic conditions, does not provide flexibility or a real-time response capability. With a system like StreetSmart, though, congestion pricing could potentially become much more sophisticated and ubiquitous. First, StreetSmart could paint a detailed picture of traffic congestion around a city, allowing transient as well as persistent trouble spots to be identified.

Some states and localities are exploring the concept of a vehicle miles tax (VMT), which would charge drivers a tax based on how many miles are driven. The idea would be to keep an OBD-II or other type of logger in the vehicle to determine the amount that each driver would pay in tax. If integrated with a system like StreetSmart, the VMT could be structured to incentivize vehicles to avoid traffic hotspots. This could be set up for known hotspots on a daily basis or for real-time hotspots that arise due to unpredictable conditions. Either way, drivers would have to be informed through the integrated VMT system and kept apprised of the real-time costs.

Of course, one major challenge of actually implementing a more sophisticated congestion tax, especially one based on crowdsourced traffic data, is system reliability and distribution. The StreetSmart system would have to be fully ironed out and proven in order to be an acceptable mechanism for driving such a controversial measure forward. Moreover, the data collection sensors could potentially be integrated with mandated VMT loggers, providing an alternative distribution method that would circumvent the challenges of launching a smartphone app.

4.3 Challenges and Limitations

Any tool with the feature set and capabilities of the StreetSmart system will come with many associated uncertainties in addition to the challenges already discussed. For instance, as researchers continue to push the boundaries of distributed and ubiquitous computing as well as data aggregation and analysis, privacy concerns will continue to rise to the forefront. This section will delve into 3 of these concerns in more detail: distribution, commercialization, and user privacy.

4.3.1 Distribution and Commercialization

The distribution of any participatory sensing system is both the most critical as well as the most challenging aspect. Initially, the data collection efforts supporting the development of StreetSmart have come from a small team of researchers at MIT. In order to support any launch of the system, however, strategies must be developed to collect both OBD-II and smartphone data from a much wider user base.

The best bet for collecting OBD-II data would be to partner directly with fleets such as taxi cabs in the city or other commercial drivers. This would allow a controlled distribution of OBD-II readers. Unfortunately, this may not allow for a diversity of vehicle types to be sampled, limiting the range of applicable vehicle regression coefficients that are available. A research effort could be made to collect a base set of data for all major vehicle makes and models in order to establish some baseline that an early system version could be built from.

There are several potential methods for distributing the StreetSmart data collection tool on smartphones. The tool can be distributed as a standalone application for the iPhone and Android platforms that users would install. However, this would most likely necessitate the simultaneous release of a traffic or energy estimation application that could be used to motivate people to download the app and contribute data. Initially, it may be easier to bundle the data collection code within another more popular application and then have the StreetSmart code running in the background.

4.3.2 Privacy

Privacy is a serious concern of many users when they are contributing personal information that may in many cases be sensitive if it is not handled with care. Each instance of the StreetSmart application, for example, will be recording location and driving data for individual users. Many users would not want their GPS coordinates and driving patterns recorded in any permanent way. The authors of [7] have worked to integrate a privacy layer into their participatory sensing network. This layer would anonymize incoming data and separate user ID's from individual road segments and pieces of data. Ultimately, once robust privacy mechanisms are demonstrated, it will be up to the users to decide whether or not to trust application developers with their personal data.

4.4 Future Work

Although we have developed the central model of StreetSmart in this thesis, we have only created a framework for the full system implementation. The next step is to build the crowdsourcing element of the project as well as develop automated tools for data processing and user management. In broad form, the most immediate future work could be outlined as follows:

1. Develop and launch the full StreetSmart crowdsourcing platform using smartphone data collection applications on iPhone and Android
2. Deploy many OBD-II / GPS logging units through fleets and volunteer vehicles
3. Launch user applications with tools for traffic avoidance, vehicle energy estimation, greenest path calculations, etc
4. Create an API enabling the development of traffic related tools based on the real-time Energy Index data
5. Implement strategies for distribution as well as privacy protection
6. Oversee the implementation of StreetSmart related applications in the areas of traffic data, congestion taxes, hybrid vehicles, electric vehicles, etc

Clearly, the first two steps are necessary in order to get the StreetSmart system to a functional level. After that, the rest are completely flexible based on which applications and research seem most promising.

4.5 Conclusions

When fully implemented, the StreetSmart system could potentially be a valuable tool to help improve the efficiency of transportation networks, especially in highly congested and urban areas. Additionally, the regression model developed in Chapter 2 represents an improvement over existing methods of predicting fuel consumption as a function of street types and traffic conditions. While current mainstream traffic reports typically include only average speed, we showed through the StreetSmart model that average speed alone is not enough to characterize traffic if the end goal is to estimate the effect on vehicle fuel consumption. Our model could be improved through additional verification work with regards to persistence, road type classification, driver behavior, and road conditions. Additionally, Although a detailed framework for the StreetSmart system was presented in Chapter 3, the full system implementation should be undertaken with the evolving needs of the most promising applications in mind.

Appendix A Selected Matlab Code

A.1 ReverseGeocoding.m

```
% function reverseGeo.m
% This script performs reverse geocoding on GPS data for one file. It also
% performs error checking and pauses if the server returns too many blank
% values. The GeoNames server only allows 5000 queries per hour and 30000
% queries per day as of 1/1/2011

function [numBlanks] = reverseGeo(fileName)

load(['RunFiles/' fileName]);        % Access stored data
lat = Trip.Data.PositionLatitude;    % GPS data
lng = Trip.Data.PositionLongitude;
n = length(lat);

try
    start = length(Trip.Data.Street1);  % If reverse geocoding exited
catch ME                                % incomplete, start from last value
    start = 1;
end

countEmpty = 0;    % If server starts returning empty values, we count them
                   % to see if there are many consecutive problems
numBlanks = 0;

for i = start:n

    [i n]
    empty = false;
    while 1
        try
            % Access GeoNames.org database
            url =
strcat('http://ws.geonames.org/findNearestIntersection?lat=',num2str(lat(i)),
'&lng=',num2str(lng(i)));
            a = urlread(url);
            break;
        catch ME
            % If database doesn't respond, our connection is bad or server
            % is down
            disp('URL not responding - Waiting 1 minute');
            pause(60);
        end
    end

    % Parse incoming data file
    numEOL = 0;
    PosEOL = zeros(8,1);
    for j = 1:length(a);
        if double(a(j)) == 10
```

```
            numEOL = numEOL + 1;
            PosEOL(numEOL) = j;
            if numEOL == 8
                break;
            end
        end
    end
    str = a(PosEOL(3)+1:PosEOL(3+1)-1);
    str = str(10:length(str)-10);

    % Database has limited our queries for the hour or the day
    if isempty(str)
        disp('waiting to get data from the server')
        countEmpty = countEmpty + 1;
        empty = true;
        numBlanks = numBlanks + 1;
    else
        % Data is good
        countEmpty = 0;
        disp('success in getting data from the server');
    end

    % only record data if we had a valid return from the server
    if ~empty
        % PosEOL contains indices of EOL characters for up to 8 EOL's
        % Street 1 information is from EOL3 to EOL4
        ind = 3;
        str = a(PosEOL(ind)+1:PosEOL(ind+1)-1);
        Street1{i,1} = str(10:length(str)-10);
        str(10:length(str)-10);

        % Street 2 information is from EOL4 to EOL5
        ind = ind+1;
        str = a(PosEOL(ind)+1:PosEOL(ind+1)-1);
        Street2{i,1} = str(10:length(str)-10);
        {Street1{i,1} Street2{i,1}}

        % Intersection Lat information is from EOL5 to EOL6
        ind = ind+1;
        str = a(PosEOL(ind)+1:PosEOL(ind+1)-1);
        IntLat(i) = str2double(str(6:length(str)-6));

        % Intersection Lng information is from EOL6 to EOL7
        ind = ind+1;
        str = a(PosEOL(ind)+1:PosEOL(ind+1)-1);
        IntLng(i) = str2double(str(6:length(str)-6));

        % Intersection Distance information is from EOL7 to EOL8
        ind = ind+1;
        str = a(PosEOL(ind)+1:PosEOL(ind+1)-1);
        IntDistance(i) = str2double(str(11:length(str)-11));
    else
        if i ~= 1 && i~=start
            Street1{i,1} = Street1{i-1,1};
            Street2{i,1} = Street2{i-1,1};
            IntLat(i) = IntLat(i-1);
            IntLng(i) = IntLng(i-1);
            IntDistance(i) = IntDistance(i-1);
        else
            Street1{i,1} = '';
            Street2{i,1} = '';
            IntLat(i) = 0;
            IntLng(i) = 0;
            IntDistance(i) = 0;
        end

        % We have received a lot of blank results...our queries are being
        % limited, wait for an hour
        if countEmpty >= 150
            disp('150 Blank results in a row.....pausing 60 minutes')
            pause(1800);
            disp('     30 mins...');
            pause(1800);
            disp('     30 mins...');
            countEmpty = 0;
        end
    end

    % Save data back to the main structure
    try
        Trip.Data.('Street1'){i,1} = Street1{i,1};
        Trip.Data.('Street2'){i,1} = Street2{i,1};
        Trip.Data.('IntLat')(i) = IntLat(i);
        Trip.Data.('IntLng')(i) = IntLng(i);
        Trip.Data.('IntDistance')(i) = IntDistance(i);

        save(['RunFiles/' fileName],'Trip')
    catch ME
        disp('Error Saving File - Retrying Data Point');
        i = i-1;
    end
end

end
```

A.2 splitSegments.m

```
% Split traffic light segments

% Run file is the raw data file in the format Rxxxx.mat that contains the
% data from an entire vehicle run

% segList is a cell array containing the id's of the segments that the
% full path contains...a roundtrip containing segments 1-3 would be
% presented as:
% {'S0001.mat','S0002.mat','S0003.mat','S0003.mat','S0002.mat','S0001.mat'}
% It is found inside the runFile in the Trip data structure
%
% directions is an array containing the directions of the segment
% drives...1 corresponds to a drive from Node 1 to Node 2. 2 corresponds to
% a drive from Node 2 to Node 1. The above drive would be represented as:
% [1 1 1 2 2 2]   This is found in the Trip data structure

function splitSegments(runFile)

    runFilePath = strcat('RunFiles/',runFile);
    disp(['Opening File: ' runFilePath]);
    load(runFilePath);       % Contains data structure 'Trip'

    iter = 1;    % This pointer iterates through main dataset
    index1 = 0;
    index2 = 0;

    % SegList and directions are input by the OBD-II user
    segList = Trip.Data.segList;
    directions = Trip.Data.directions;

    % Iterate through all of the segments in this commute
    for i=1:length(segList)

        % Download Street Info Data About This Segment
        segFile = strcat('SegFiles/',segList{i});
        disp(['Opening File: ' segFile]);
        load(segFile);
        Str = Info.StrName;
        Node1Int = Info.Node1.IntStr;    % Intersection Streets for this seg
        Node2Int = Info.Node2.IntStr;

        revgeoStr1 = Trip.Data.Street1;      % Reverse Geocoded Values
        revgeoStr2 = Trip.Data.Street2;
        IntDist = Trip.Data.IntDistance;

        % Direction is from Node 1 to Node 2
        if(directions(i) == 1)
            if index2 == 0 || directions(i)~=directions(i-1)  % If direction
is reversing, search for starting point of next seg (i.e. don't assume it's
continuous)
                % Find Starting Index
                [index1 iter] = searchStr(iter, revgeoStr1, revgeoStr2, Str,
Node1Int, IntDist);
            else
                index1 = index2+1;          % First datapoint of this segment
equals next available datapoint after last segment
            end
            [index2 iter] = searchStr(iter, revgeoStr1, revgeoStr2, Str,
Node2Int, IntDist);
        % Direction is from Node 2 to Node 1
        else
            if index2 == 0 || directions(i)~=directions(i-1)
                % Find starting index
                [index1 iter] = searchStr(iter, revgeoStr1, revgeoStr2, Str,
Node2Int, IntDist);
            else
                index1 = index2+1;          % First datapoint of this segment
equals next available datapoint after last segment
            end
            [index2 iter] = searchStr(iter, revgeoStr1, revgeoStr2, Str,
Node1Int, IntDist);
        end


        try
            j = length(SegData)+1;
        catch ME         % Data does not exist, start at index 1
            disp('Creating new SegData struct');
            j = 1;
        end

        % Fill Out SegData Data Structure
        SegData{j}.TripTime = Trip.TripTime;
        SegData{j}.Route = Trip.Route;
        SegData{j}.TripDate = Trip.TripDate;
        SegData{j}.CarModel = Trip.CarModel;
        SegData{j}.AdditionalInfo = Trip.AdditionalInfo;
        SegData{j}.Direction = directions(i);
        SegData{j}.Driver = Trip.Driver;

        SegData{j}.Data.SampleTime = Trip.Data.SampleTime(index1:index2);
        SegData{j}.Data.PositionLatitude =
Trip.Data.PositionLatitude(index1:index2);
        SegData{j}.Data.PositionLongitude =
Trip.Data.PositionLongitude(index1:index2);
        try
            SegData{j}.Data.FuelRate1 = Trip.Data.FuelRate1(index1:index2);
        catch ME
        end
        try
            SegData{j}.Data.FuelRate2 = Trip.Data.FuelRate2(index1:index2);
        catch ME
        end
        SegData{j}.Data.VehicleSpeed = Trip.Data.VehicleSpeed(index1:index2);

        SegData{j}.runFile = runFile;
        SegData{j}.runFileIndex1 = index1;
        SegData{j}.runFileIndex2 = index2;

        % Save SegData in master data structure
        save(segFile,'Info','SegData');

        clear SegData    % clear SegData for next loop
    end

end

 % Locally defined function to find the appropriate ending and beginning
    % indices of a light-light segment based upon the reverse geocoding
    % data. Relies on finding minimum value of intDist variable for a
    % particular intersection
    function [minIndex ptr] = searchStr(ptr, geo1, geo2, str1, str2, intDist)

        str1 = upper(str1);      % Upper case to minimize string mismatch
        str2 = upper(str2);
        minIndex = length(geo1);

        while ptr<length(geo1)

            % Test to see if the reverse geo data is at the intersection
            if strcmpi(geo1{ptr}, str1) && strcmpi(geo2{ptr},
str2)||(strcmpi(geo2{ptr},str1) && strcmpi(geo1{ptr},str2))
                % Find the range of indices for which this is the nearest
                % intersection
                low = ptr;      % Low index
                ptr = ptr+1;
                while (strcmpi(geo1{ptr}, str1) && strcmpi(geo2{ptr},
str2)||(strcmpi(geo2{ptr},str1) && strcmpi(geo1{ptr},str2))) &&
ptr<length(geo1)
                    ptr = ptr+1;
                end
                high = ptr-1;    % High index

                % Set the intersection index as the min IntDist value in
                % this range
                [minDist minIndex] = min(intDist(low:high));
                minIndex = minIndex + low;
                break;
            end

            ptr = ptr+1;
        end
    end
```

A.3 defineMovSegs.m

```
% Separates Moving and Idling Components of a Segment Traversal. Stores
% transition indices as part of the Trip Data Structure as Trip.MovSegs
function [Trip] = defineMovSegs(Trip)

    % Length of time in seconds that a moving segment must
    % be in order to qualify as a full segment
    Tmin=3;

    time = Trip.Data.SampleTime;
    speed = Trip.Data.VehicleSpeed;
    iter = 1;
    segCount = 1;

    % Identify segments containing motion (filter out idling segments)
    while(iter<=length(time))
        while(iter<=length(time)&&speed(iter)==0)
            iter=iter+1;
        end
        segStart=iter-1; % start at zero velocity
        MovSegs(segCount,1)=max(segStart,1);

        while(iter<=length(time))
            if(speed(iter)==0)
                if(time(iter)-time(MovSegs(segCount,1)))<Tmin
                    break;
                else
                    MovSegs(segCount,2)=iter;
                    segCount=segCount+1;
                    break;
                end
            end
            iter=iter+1;
        end
    end

    % If only 1 seg, prevent second index from being zero or null
    [rMS cMS]=size(MovSegs);
    try
        if(MovSegs(rMS,2)==0)
            MovSegs(rMS,2)=length(time);
        end
    catch ME
        MovSegs(rMS,2)=length(time);
    end

    % Test to make sure the last segment is long enough
    if(time(MovSegs(rMS,2))-time(MovSegs(rMS,1)))<Tmin
        MovSegs = MovSegs(1:rMS-1,:);
    end

    Trip.MovSegs = MovSegs;
end
```

A.4 calcMetrics.m

```
% Calculates Energy Indices for a run, calls defineMovSegs() function and
% computeMetrics() function.

function calcMetrics()

load('SegFiles/SegVars.mat');

% Iterate through Segments that haven't had calculated metrics yet
for seg=1:numSegs
    filename=strcat('S',num2str(seg,'%04.0f'),'.mat');
    load(['SegFiles/' filename]);
    disp(['Calculating Metrics for ' filename '...']);

    for j=1:length(SegData)
        if metcalc(j)==0    % Check if metrics have been calculated yet

            % Split segment into moving and idle portions
            SegData{j} = defineMovSegs(SegData{j});

            %Compute metrics for moving portions
            SegData{j} = computeMetrics(SegData{j});

            % Set metric flag to 1, indicating metrics have been computed
            metcalc(j) = 1;

        end
    end

    % Save master data structure
    save(['SegFiles/' filename], 'SegData','Info','metcalc');

end
end

% Compute the energy indices of a provided Trip data structure
function [Trip] = computeMetrics(Trip)

    Tsample = 0.25; % Compute metrics at 4 Hz

    % First get the raw data for the run
    time = Trip.Data.SampleTime;
    speed = Trip.Data.VehicleSpeed;
    MovSegs = Trip.MovSegs;

    % Determine which OBD-II fuel sensor data is available
    try
        fcrate = Trip.Data.FuelRate2;
    catch ME
        try
            fcrate = Trip.Data.FuelRate;
        catch ME
```

```
            disp(['No Fuel Data present in: ' Trip.runFile]);
            return;
        end
    end

    [rMS cMS] = size(MovSegs);

    for seg=1:rMS

        StartPos = MovSegs(seg,1);
        EndPos = MovSegs(seg,2);

        % First, generate uniformly sampled data at specified sample rate
        [tempTime tempSpeed] =
suppressRep(time(StartPos:EndPos),speed(StartPos:EndPos));
        uniformTime = floor(min(tempTime))+1:Tsample:floor(max(tempTime));

        % Convert speed to m/s from mph
        uniformSpeed = interp1(tempTime,tempSpeed,uniformTime)*.44704;

        % Log Start time for segment
        timeStr = Trip.TripTime;
        StartTime = str2num(timeStr(1:2))+str2num(timeStr(4:5))/60;
        StartTime = StartTime + time(StartPos)/3600;

        %Commute Time Metric
        DriveTime = time(EndPos)-time(StartPos);

        % Calculate acceleration for uniform velocity data points
        uniformAcc = zeros(length(uniformTime),1);
        uniformAcc(1)=0;
        for n=2:length(uniformAcc)
            uniformAcc(n)=(uniformSpeed(n)-uniformSpeed(n-1))/Tsample;
        end

        % Calculate distance from uniform velocity
        uniformDist = zeros(length(uniformTime),1);
        uniformDist(1)=0;
        for n=2:length(uniformDist)
            uniformDist(n)=uniformDist(n-
1)+0.5*(uniformSpeed(n)+uniformSpeed(n-1))*Tsample;
        end

        % Calculate v2 and v3 values
        vsq = zeros(length(uniformTime),1);
        vcub = zeros(length(uniformTime),1);
        for n=1:length(vsq)
            vsq(n) = uniformSpeed(n)^2;
            vcub(n) = uniformSpeed(n)^3;
        end

        iadt = 0;
        ivdt = 0;
        ivsqdt = 0;
        ivcubdt = 0;
        for n=2:length(uniformTime)
            iadt = iadt + abs(uniformSpeed(n)-uniformSpeed(n-1));
            ivdt = ivdt + Tsample*((uniformSpeed(n)+uniformSpeed(n-1))*.5);
            ivsqdt = ivsqdt + Tsample*((vsq(n)+vsq(n-1))*.5);
            ivcubdt = ivcubdt + Tsample*((vcub(n)+vcub(n-1))*.5);
        end

        iadx = 0;

        for n=2:length(uniformDist)
            iadx = iadx + abs((uniformSpeed(n)-uniformSpeed(n-
1))/Tsample)*(uniformDist(n)-uniformDist(n-1));
        end

        % avgspeed = uniformDist(length(uniformDist)) /
        %   uniformTime(length(uniformTime));
        avgspeed = mean(uniformSpeed);

        % Distance driven for segment
        dist = ivdt;  % These are the same!!

        % Calculate Fuel Consumption
        fctot=0;
        for n=StartPos+1:EndPos
           fctot = fctot + fcrate(n)*(time(n)-time(n-1))/3600;
        end

        Trip.Metric.iadt(seg)=iadt;
        Trip.Metric.iadx(seg)=iadx;
        Trip.Metric.ivdt(seg)=ivdt;
        Trip.Metric.ivsqdt(seg)=ivsqdt;
        Trip.Metric.ivcubdt(seg)=ivcubdt;
        Trip.Metric.StartTime(seg)=StartTime;
        Trip.Metric.drivetime(seg)=DriveTime;
        Trip.Metric.fc(seg)=fctot;
        Trip.Metric.segLength(seg)=dist;
        Trip.Metric.avgspeed(seg)=avgspeed;
    end

    % Store uniform speed and time as metrics for future reference
    [tempTime tempSpeed] = suppressRep(time,speed);
    uniformTime = floor(min(tempTime))+1:Tsample:floor(max(tempTime));

    % Convert speed to m/s from mph
    uniformSpeed = interp1(tempTime,tempSpeed,uniformTime)*.44704;
    Trip.Data.uniformTime = uniformTime;
    Trip.Data.uniformSpeed = uniformSpeed;

    % Calculate total idle time for segment
    idletime = 0;
    for i=2:length(time)-1
        if speed(i) == 0
            deltaT1 = time(i)-time(i-1);
            deltaT2 = time(i+1)-time(i);
            idletime = idletime + .5*(deltaT1 + deltaT2);
        end
    end
    Trip.Metric.idletime = idletime;

    %Calculate total idle fuel consumption
    idlefuel = 0;
    fctot = 0;
    for i=2:length(time)
        fctot = fctot + fcrate(i)*(time(i)-time(i-1))/3600;
    end
    Trip.Metric.fctot = fctot;
    Trip.Metric.fcidle = fctot - sum(Trip.Metric.fc);

end

% This code suppresses repetitions in the input data.
% It takes a input time array and velocity value array
% it outputs new time array and new velocity array where there are no
% repetitions in the t values.
function [newt,newV] = suppressRep(t,V)

    count = 2;
    newV(1,1) = V(1);
    newt(1,1) = t(1);
    for i = 2:length(t)
        if t(i) ~= t(i-1)
            newt(count,1) = t(i);
            newV(count,1) = V(i);
            count = count + 1;
        end
    end
end
```

Bibliography

[1] Bureau of Transportation Statistics. [Online]. http://www.bts.gov

[2] "Annual Review 2009," U.S. Energy Information Administration, 2010.

[3] Joshua Siegel, "Design, Development, and Validation of a Remotely Reconfigurable Vehicle Telemetry System for Consumer and Government Applications," Massachusetts Institute of Technology, Thesis.

[4] OBD-II. [Online]. http://www.obdii.com/background.html

[5] Arvind Thiagarajan et al., "VTrack: accurate, energy-aware road traffic delay estimation using mobile phones," in *SenSys*, 2009, pp. 85-98.

[6] Brett Hull et al., "CarTel: a distributed mobile sensor computing system," in *SenSys*, 2006, pp. 125-138.

[7] Raghu K. Ganti, Nam Pham, Hossein Ahmadi, Sauabh Nangia, and Tarek F. Abdelzaher, "GreenGPS: a participatory sensing fuel-efficient maps application," in *MobiSys*, 2010, pp. 151-164.

[8] Juan C. Herrera, Daniel B. Work, Ryan Herring, Xuegang Ban, and Alexandre M. Bayen, "Evaluation of Traffic Data Obtained via GPS-Enabled Mobile Phones: the Mobile Century Field Experiment," UC Berkeley Center for Future Urban Transport, Working Paper 2009.

[9] Tim Kaler et al., "Demo Abstract: Code in the Air - Simplifying Sensing on Smartphones," in *SenSys*, Zurich, 2010, pp. 407-408.

[10] Navteq - Traffic.com. [Online]. http://www.traffic.com

[11] INRIX - go anywhere. [Online]. http://www.inrix.com/

[12] Airsage. [Online]. http://www.airsage.com

[13] Gayathri Chandrasekaran et al., "Vehicular speed estimation using received signal strength from mobile phones," in *Ubicomp*, 2010, pp. 237-240.

[14] Shravan Gaonkar, Jack Li, Romit Roy Choudhury, Landon Cox, and Al Schmidt, "Micro-Blog: sharing and querying content through mobile phones and social participation," in *MobiSys*, 2008, pp. 174-186.

[15] R. Herring et al., "Using Mobile Phones to Forecast Arterial Trafic Through Statistical Learning," 89th Transportation Research Board Annual Meeting.

[16] Prashanth Mohan, Venkata N. Padmanabhan, and Ramachandran Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *SenSys*, 2008, pp. 323-336.

[17] D.B. Work and A.M. Bayen, "Impacts of the Mobile Internet on Transportation Cyberphysical Systems: Traffic Monitoring using Smartphones," in *National Workshop for Research on High-Confidence Transportation Cyber-Phyiscal Systems: Automotive, Aviation and Rail*, November 2008.

[18] D.B. Work, O.P. Tossavainen, and A. Bayen Jacobson, "Lagrangian Sensing: Distributed Traffic Estimation with Mobile Devices," in *American Control Conference*, 2009, pp. 1536-1543.

[19] D.B. Work, S. Blandin, B. Piccoli, and A. Bayen, "A traffic model for velocity data assimilation," Applied Mathematics Research eXpress, 2010.

[20] Jungkeun Yoon, Brian Noble, and Mingyan Liu, "Surface street traffic estimation," in *MobiSys*, 2007, pp. 220-232.

[21] Ali Emadi, Yimin Gao, and Mehrdad Ehsan, *Modern Electric, Hybrid Electric, and Fuel Cell Vehicles: Fundamentals, Theory, and Design*, 2nd ed.: CRC Press, 2010.

[22] D.Y.C. Leung and D.J.C. Williams, "Modeling of motor vehicle fuel consumption and emissions using a power-based model," *Environmental Monitoring and Assessment*, vol. 65, no. 1-2, pp. 21-29, 2000.

[23] Eric Jackson, "Improving traffic simulation models and emissions models using ob-board vehicle dynamics data," University of Connecticut, PhD Thesis 2008.

[24] P.G. Boulter, I.S. McCrae, and T.J. Barlow, "A review of instantaneous emission models for road vehicles," Transport Research Laboratory Project Reports, 2006.

[25] Donnie H. Kim, Younghun Kim, Debora Estrin, and Mana B. Srivastava, "SensLoc: sensing everyday places and paths using less energy," in *SenSys*, 2010, pp. 43-56.

[26] Google Real-Time Traffic. [Online]. http://maps.google.com/support/bin/answer.py?answer=61454

[27] AVL ADVISOR. [Online]. https://www.avl.com/produkte

[28] Kaisen Lin, Aman Kansal, Dimitrios Lymberopoulos, and Feng Zhao, "Energy-accuracy trade-off for continuous mobile device location," in *MobiSys*, 2010, pp. 285-298.

[29] Autotap reader. [Online]. http://www.autotap.com/products.asp

[30] AutoXRay. Ez-scan. [Online]. http://www.autoxray.com/productcategory.php?id=338

[31] Auterra Dash Dyno. [Online]. http://www.auterraweb.com/dashdynoscantool.html

[32] ZipCar. [Online]. http://www.zipcar.com

[33] Joe Khoury, "Traffic Characterization and Road Categorization," Massachusetts Institute of Technology, Thesis 2010.

[34] GeoNames. [Online]. http://www.geonames.org

[35] (2006, June) MultiChannel Merchant: Fuel Conservation No Idle Matter at UPS. [Online]. http://multichannelmerchant.com/opsandfulfillment/advisor/fuel_conserve/

[36] M.B. Kjaergaard, J. Langdal, T. Godsk, and Toftkjaer, "EnTracked: Energy-Efficient Robust Position Tracking for Mobile Devices," in *MobiSys*, 2009, pp. 221-234.

[37] I. Skog and P. Handel, "In-Car Positioning and Navigation Technologies - A Survey," *IEEE Transactions On Intelligent Transportation Systems*, pp. 4-21, 2009.

[38] Y. H. Hong, H.K. Kim, and H.S. Lee, "Reconstruction of dynamic displacement and velocity from measured accelerations using the variational statement of an inverse problem," *Journal of Sound and Vibration*, pp. 4980-5003, 2010.

[39] L. Ojeda and J. Borenstein, "Non-GPS Navigation with the Personal Dead-Reckoning System," in *SPIE*, 2007.

[40] Suhas Mathur et al., "ParkNet: drive-by sensing of road-side parking statistics," in *MobiSys*, 2010, pp. 123-136.

[41] Susanne Breitenberger, "XFCD - Extended Floating Car Data Potentials and Penetration Rates," BMW Group,.

[42] Jeffrey Gonder and Tony Markel, "Energy Management Strategies for Plug-In Hybrid Electric Vehicles," in *SAE World Congress*, Detroit, 2007.

[43] Qiuming Gong, Yaoyu Li, and Zhong-Ren Peng, "Trip-Based Optimal Power Management of Plug-in Hybrid Electric Vehicles," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 6, pp. 3393-3401, November 2008.

[44] Georgia-Evangelia Katsargyri, "Optimally Controlling Hybrid Electric Vehicles using Path Forecasting," Massachusetts Institute of Technology, Master's Thesis 2008.