Wolverine: Traffic and Road Condition Estimation using Smartphone Sensors

Ravi Bhoraskar

May 4, 2012

under guidance of,
Prof. Bhaskaran Raman,
IIT Bombay

Outline

Introduction

- Growing population : Growing Number of vehicular users
- Growing vehicular users : Growing traffic
- Need a mechanism to estimate traffic

Traffic in Mumbai [3]

Traffic in Hyderabad [2]

Problem Statement

To design a complete, scalable smartphone based system to sense, infer and present road traffic and road state information in real time

Summary of Work Done I

- Studied some of the previous work
- Learned about the sensors in smartphones and the Android API
- Developed a **Virtual Reorientation algorithm**
- Developed a Machine Learning technique to identify bump and braking events
- Evaluation of the algorithm with data collected on roads of IIT-Bombay campus and Hiranandani, Powai
- Energy consumption model, and comparision with an existing approach
- Paper accepted at **WISARD-2012** workshop

Summary of Work Done II

- Developed an application that implements these algorithms online
- Developed an application to present the traffic scenario on mobile
- Studied ways to implement inter-phone communication
- Devised algorithms to find if devices are in the same vehicle

Division of Work

Joint project with Nagamanoj Vankadhara. The division of work was as follows

- Event Detection on Smartphone
  - Virtual Reorientation Algorithm: *Nagamanoj*
  - Machine Learning Algorithms for Event Detection: *Ravi*
  - Energy Consumption Model: *Ravi*
  - Android App: *Ravi and Nagamanoj*
  - Experimental Evaluation: *Ravi and Nagamanoj*
- Inter-Phone Communication
  - Establishing Wireless Link: *Nagamanoj*
  - Distance Metrics: *Ravi and Nagamanoj*
- Presentation on Phone: *Ravi*

Related Work

Method	Interesed in	Hardware	Scalability	Accuracy
Auto Witness [7]	Vehicle Trajectory	Accelerometer, Gyro, GSM	No	>90%
Pothole Patrol [6]	Road State detection	Accelerometer, GPS	No	$<$ 0.2% false positives
Road Sound Sense [9]	Vehicle speed	Acoustic sensors	No	accuracy varying b/w 85.7% to 100%
Nericell [8]	Vehicle acceleration	Accelerometer, Microphone, GPS, GSM Antenna	Yes	11.1% false positives and 22% false negatives
Wolverine (Our Method)	Vehicle acceleration	Accelerometer, Magnetometer, GPS	Yes	-

Table: Comparison among some of the previous methods and our method

Overview of System

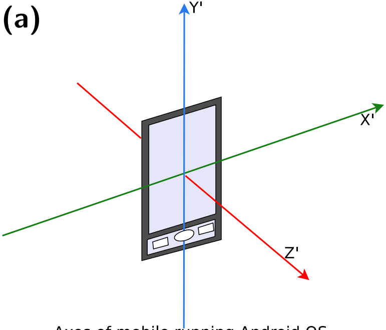
Figure: Wolverine System Schematic

Virtual Reorientation: Need

- Phone can be placed arbitrarily with respect to the vehicle
- Event detection algorithms work on acceleration from X, Y and Z axes differently (detect braking as bump in vertical phone)
- Phone orientation can change even during vehicle motion
- Hence, continuous virtual reorientation is required

Framework

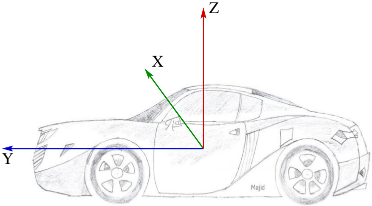
(a)



Axes of mobile running Android OS

Phone's axes

(b)



Vehicle's axes

Reorientation in Nericell

1. Wait till angle with X-Y plane changes, to trigger reorientation
2. Use accelerometer to compute angle with X-Y plane
3. Turn on GPS, and wait for braking event. Use $\vec{a_Y} = \vec{a} - \vec{a_Z}$ to compute Y

Reorientation in Wolverine

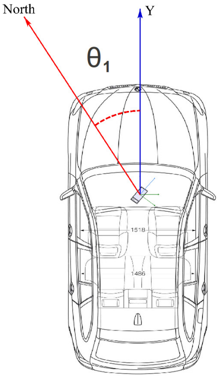


Figure: Calculating Bearing

1. Find angle with X-Y plane, like Nericell
2. Find angle with north, in phone coordinates
3. Calculate direction of motion using GPS
4. Find angle with north, in vehicle coordinates
5. Subtract the vectors to find the bearing of phone w.r.t. vehicle

Machine Learning Algorithms: Motivation

- Nericell uses fixed thresholds on accelerometer values
- Threshold may vary across vehicles, road conditions and the mobile device
- Let the thresholds be learned, for better performance

Overview of the Technique

1. First, reorient the accelerometer data
2. Divide into 1 second windows
3. Extract features from each window
4. Use k-means on the training data, then label it
5. Use labeled data to train SVM
6. Classify the incoming data using SVM

Features considered

The features that we considered to extract from the accelerometer data were

- Mean (μ)
- Standard Deviation (σ)
- Max $-$ Min over the window (δ)

$$\delta_X = max_{a_i \in window} a_i - min_{a_i \in window} a_i$$

Training for Bump Events

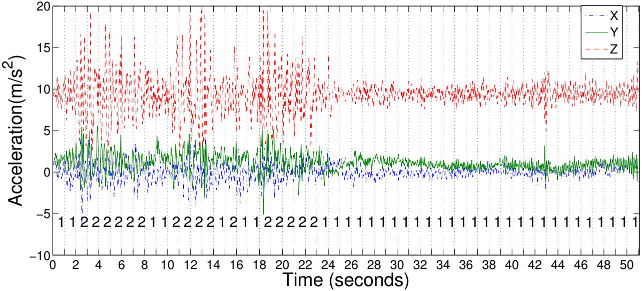


Figure: Accelerometer data for bumpy(0-25s) and smooth(25-51s) road

Training for Bump Events

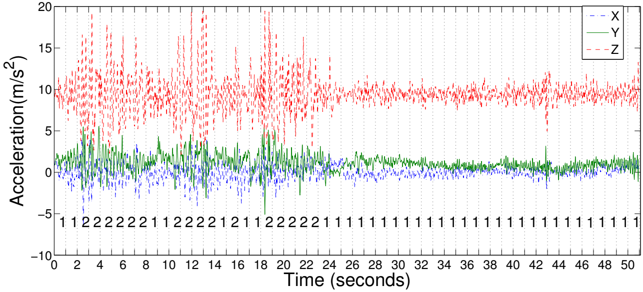


Figure: Accelerometer data for bumpy(0-25s) and smooth(25-51s) road

We choose σ_Z as the only feature for detecting bumps

Classification of Bump Events

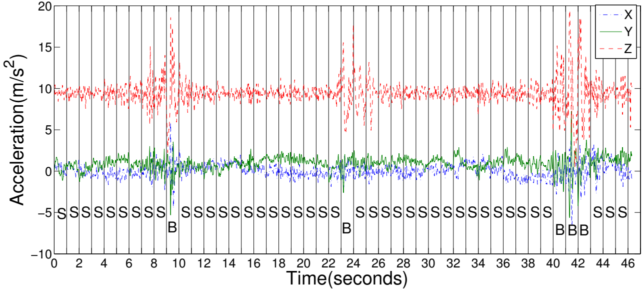


Figure: Accelerometer Data for three speedbreakers

All the *B's* are the bump events. The algorithm correctly identified three speadbreakers

Training for Braking Events

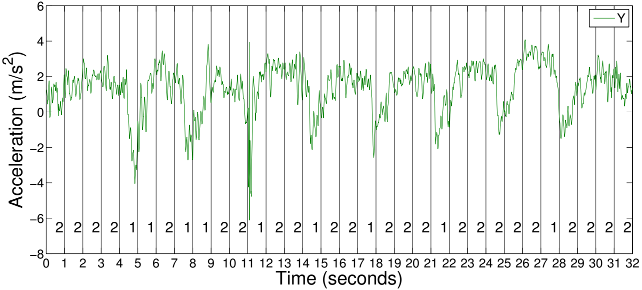


Figure: Y axis Accelerometer for braking events(with labels)

Training for Braking Events

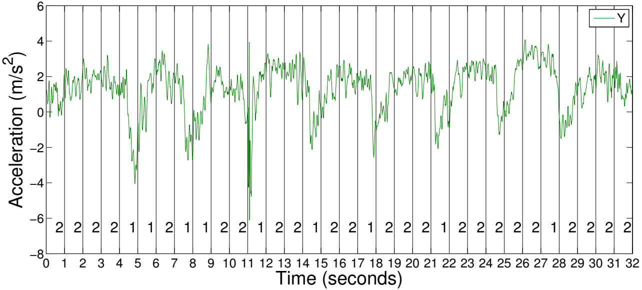


Figure: Y axis Accelerometer for braking events(with labels)

We choose δ_Y as the only feature for detecting bumps

Classification of Braking Events

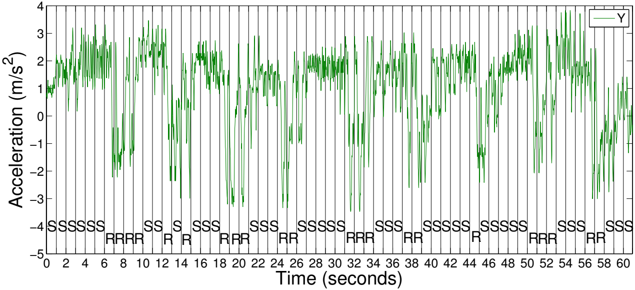


Figure: Braking events with generated class labels

The *R's* are the braking events. 9 events are detected

Experimental Setup

- Used **HTC Wildfire S**, **Samsung Nexus S** and **LG Optimus**
- Both running **Android OS 2.3.3** (Gingerbread) and SDK Version 10
- Both have **accelerometer, magnetometer** and **GPS sensors**
- Nexus S has **gyroscope** as well (did not use this for now)
- Collected data on *Suzuki Access 125* and *Bajaj Autorickshaw* in IIT-Bombay campus and in Hiranandani, Powai

Controlled Dataset Experiment: Bump Training

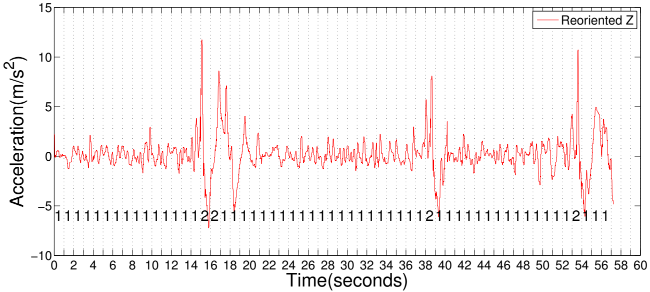


Figure: Accelerometer readings in reoriented Z-direction for scooter training data, with clusters

Controlled Dataset Experiment: Bump Testing

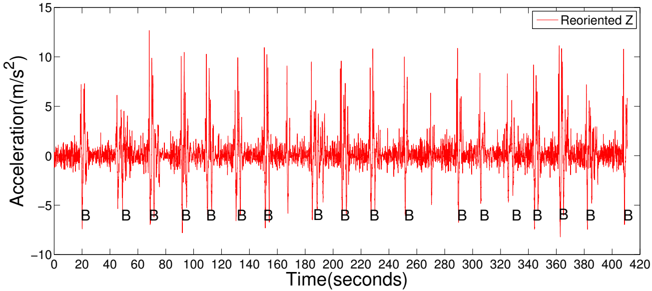


Figure: Accelerometer readings in reoriented Z-direction for scooter test data, with labels

Correctly identified 18 out of 20 bump events

Bump Detection Experimental Results

	False Positives	False Negatives
Scooter	0 %	10 %
Auto Rickshaw	8 %	0 %

Braking Detection Experimental Results

	False Positives	False Negatives
Scooter	2.7 %	21.6 %
Auto Rickshaw	0 %	13.1 %

Real Traffic Dataset: Bump Detection

Run No.	Events Detected	False Positives	FP %	False Negatives	FN %
1.	21	0	0	1	4.76
2.	23	4	17.39	1	4.34
3.	7	0	0	0	0
4.	50	0	0	0	0
Total	101	4	3.96	2	1.98

Table: Bump Detection Results

Read Traffic Dataset: Braking Detection

Run No.	Events Detected	False Positives	FP %	False Negatives	FN %
1.	88	8	9.09	2	2.27
2.	59	13	22.03	0	0
3.	50	7	14	0	0
Total	197	28	14.21	2	1.01

Table: Braking Detection Results

Evaluation Summary

- Bump Detection gets very low FP and FN
- Braking Detection gets *either* low FP or low FN
  - **Hypothesis**: Depends on the training data
- Good training data is very important to get good results
- Training on bus or other public transport is still a challenge

Outline of Energy Consumption Model

- Identify major points of battery drain, and create model
- Compare Wolverine with Nericell, and quantify energy savings

Outline of Energy Consumption Model

- Identify major points of battery drain, and create model
- Compare Wolverine with Nericell, and quantify energy savings

- GPS, Acclerometer, Magnetometer and CPU consume energy
- CPU power consumption very low, hence ignored
- Accelerometer, Magnetometer always on, hence constant energy
- GPS energy consumption is interesting

GPS On Time

- Wolverine

$$t = (\text{TTFF} + \text{Time to record}) \times 2$$

$$t = \text{TTFF} + \text{Time to record} \times 2 + \text{Vehicle Motion Time}$$

- Nericell

$$t = \text{TTFF} + \text{Time till braking} + \text{Time for braking}$$

Parameters of the Energy Model

Activity	Time
TTFF	5.5s
Time to Record	0
Time till braking	60s
Time for braking	2s
Vehicle Move Time	2s

Table: The parameters of energy model

Energy Consumption

Modality	Power Consumed	Time On (Nericell)	Time On (Wolverine)
GPS	$617.3mW$	10 %	1.1 %
Sensors + CPU	$31.85mW$	100 %	100 %

Table: Energy Consumption [4] [8] [5]

Energy Savings compared to Nericell

- Per reorientation event

$$1 - \frac{5 + 0 \times 2 + 2}{5 + 60 + 2} = 89\%$$

- Total Energy Saved

$$1 - \frac{0.11 \times 617.3 \times 0.1 + 31.85}{617.3.1 + 31.85} = 58\%$$

App for Online Event Detection

Demo!

Inter-Phone communication: Motivation

Consider 2 phones running Wolverine on the same vehicle. Same events are detected by both phones

- Energy can be conserved if we switch off the sensors on one phone.
- Will uduly bias the inference algorithm, since 2 vehicles considered instead of one

Inter-Phone communication: Motivation

Consider 2 phones running Wolverine on the same vehicle. Same events are detected by both phones

- Energy can be conserved if we switch off the sensors on one phone.
- Will uduly bias the inference algorithm, since 2 vehicles considered instead of one
- Phones in vicinity locally communicate, and infer whether in same vehicle
- Time-share the sensing between the devices

Establishing Wireless Link

- In general, a multihop P2P protocol; we consider 2 phones only
- Phones in communication range establish link and exchange files using
  - Bluetooth
  - WiFi
- Infer whether phones are in same vehicle or not

Determining if phones are in same vehicle

- Exchange the traces of braking events for last 30 seconds (0-1 string)
- Calculate a “distance” between the two strings
- Conclude that the devices are in same vehicle if the distance is below a threshold

The Distance Metric

1. The distance between two identical strings should be zero
2. The distance between a string of all 1's and a string of all 0's should be the maximum possible distance
3. Two strings with greater number of mismatches should have a greater distance than two strings with lesser number of mismatches. For example $D(111, 110) > D(111, 100)$
4. If there is a mismatch of 1's in the strings, the distance should be more in case the 1's are more far apart in the two strings. For example $D(100, 001) > D(100, 010)$
5. If two strings can be matched by shifting one of the strings, then the distance should be low. However, the distance must be higher when a higher number of shifts are required. $D(11010000, 11010000) < D(11010000, 01101000) < D(11010000, 00110100)$

Distance Metrics I

For two strings $X[0 \dots N]$ and $Y[0 \dots N]$

- Euclidean Distance

$$d_{Euclidean}(X, Y) = \sqrt{\sum_{n=1}^{N} (X[n] - Y[n])^2}$$

- Hamming Distance

$$d_{Hamming}(X, Y) = \sum_{n=1}^{N} (X[n] \oplus Y[n])$$

Distance Metrics II

- **Levenshtein Distance** The minimum number of edits needed to transform one string into the other, with the allowable edit operations being insertion, deletion, or substitution of a single character [1]. A well known dynamic programming algorithm is used
- **Shifting Window Euclidean**

$$d_{SWE}(X,Y) = \min_{k=1}^{N}\left(\sqrt{\sum_{n=1}^{k} X[i]^2 + \sum_{n=k+1}^{N-k}(X[i]-Y[i+k])^2 + \sum_{n=N-k+1}^{N} Y[i]^2 + p(k)}\right)$$

where p is the penalty function.

- **Average Distance from Closest 1**

$$d_{ADC1} = \frac{\sum_{(1\le n\le N)\wedge X[i]=1}\left(\min_{(1\le m\le N)\wedge Y[m]=1}|m-n|\right) + \sum_{(1\le n\le N)\wedge Y[i]=1}\left(\min_{(1\le m\le N)\wedge X[m]=1}|m-n|\right)}{\sum_{n=1}^{N}(X[i]=1) + \sum_{n=1}^{N}(Y[i]=1)}$$

Distance Metrics III

Presentation on Phone

Demo!

Conclusions and Future Work

- Conclusions
  - Considering the use of magnetometer reduces the energy in reorientation of accelerometer axes
  - Traffic and road state estimation is possible, by braking and bump detection respectively
  - Scalable system, as any user having smartphone can participate

Conclusions and Future Work

- Conclusions
  - Considering the use of magnetometer reduces the energy in reorientation of accelerometer axes
  - Traffic and road state estimation is possible, by braking and bump detection respectively
  - Scalable system, as any user having smartphone can participate
- Future Work
  - Fully implement application that can be installed by the users in their smartphones
  - Design the server-side traffic inference algorithm
  - Implement and test the inter-phone communication
  - Localization in energy efficient manner
  - Differentiating classes of vehicles

References I

- Levenshtein distance.
  http://en.wikipedia.org/wiki/Levenshtein_distance.
- Traffic in hyderabad on a typical day.
  http://www.hindu.com/thehindu/gallery/0463/046302.jpg.
- Traffic in mumbai on a typical day.
  http://my.opera.com/bentrein/albums/showpic.dml?album=667375&picture=9790309.
- M. Amir Yosef.
  Energy-aware location provider for the android platform.
  Master's thesis, University of Alexandria, Egypt, 2010.

References II

A. Carroll and G. Heiser.
An analysis of power consumption in a smartphone.
In *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, USENIXATC'10, pages 21–21, Berkeley, CA, USA, 2010. USENIX Association.

J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan.
The pothole patrol: Using a mobile sensor network for road surface monitoring.
In *The Sixth Annual International conference on Mobile Systems, Applications and Services (MobiSys 2008)*, Breckenridge, U.S.A., June 2008.

References III

S. Guha, K. Plarre, D. Lissner, S. Mitra, B. Krishna, P. Dutta, and S. Kumar.
Autowitness: locating and tracking stolen property while tolerating gps and radio outages.
In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 29–42, New York, NY, USA, 2010. ACM.

P. Mohan, V. N. Padmanabhan, and R. Ramjee.
Nericell: rich monitoring of road and traffic conditions using mobile smartphones.
In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 323–336, New York, NY, USA, 2008. ACM.

References IV

R. Sen, P. Siriah, and B. Raman.
Roadsoundsense: Acoustic sensing based road congestion monitoring in developing regions.
In *SECON*, pages 125–133, 2011.

That's All Folks!

Questions?