Wolverine: Traffic and Road Condition Estimation using Smartphone Sensors

Ravi Bhoraskar

November 30, 2011

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 [2]

Traffic in Hyderabad [1]

Problem Statement

- Design a smart phone based solution
- Traffic estimation: free flowing vs congested braking
- Road conditions and anomalies: smooth vs bumpy
  bumps and potholes
- Differentiate class of the vehicle: two wheeler, three wheeler or four wheeler patterns in the acceleration values along different axes

Summary of Work Done

- 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
- Developed an application to collect data from sensors to test reorientation algorithm
- Basic evaluation of the algorithm with data collected on roads of IIT-Bombay campus
- Energy consumption model, and comparision with an existing approach
- Paper accepted at **WISARD-2012** workshop

Division of Work

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

- Nagamanoj
  - Virtual Reorientation algorithm
- Ravi
  - Machine Learning algorithm
  - Energy Consumption model
- Both
  - Android application for data collection
  - Experimental evaluation

Related Work

Method	Interesed in	Hardware	Scalability	Accuracy
Auto Witness [6]	Vehicle Trajectory	Accelerometer, Gyro, GSM	No	>90%
Pothole Patrol [5]	Road State detection	Accelerometer, GPS	No	< 0.2% false positives
Road Sound Sense [8]	Vehicle speed	Acoustic sensors	No	accuracy varying b/w 85.7% to 100%
Nericell [7]	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	-

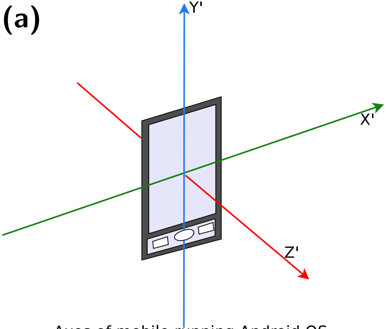
Table: Comparison among some of the previous methods and our method

Need for reorientation

- 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

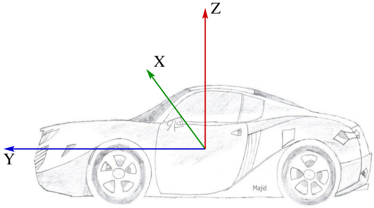
(a)



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

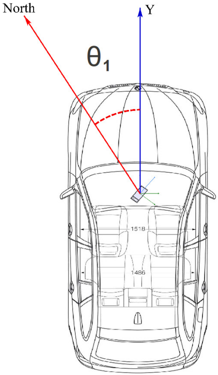


Figure: Calculating Bearing

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

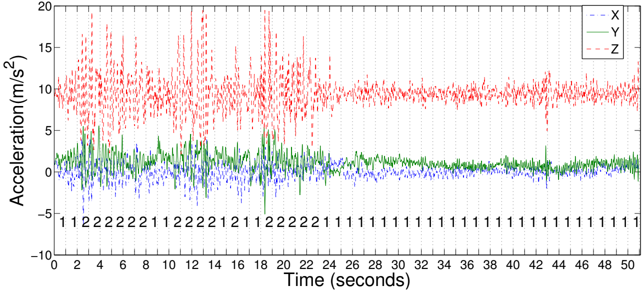


Figure: Accelerometer data for bumpy(0-25s) and smooth(25-51s) road

Training for Bump Events

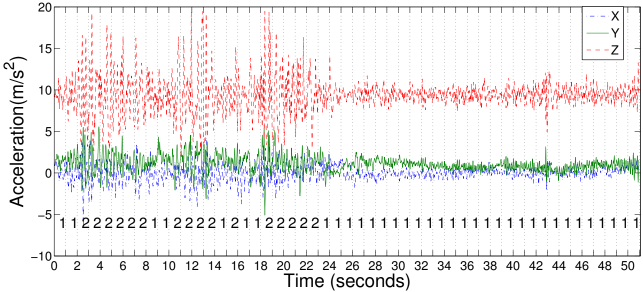


Figure: Accelerometer data for bumpy(0-25s) and smooth(25-51s) road

We choose μ_Z as the only feature for detecting bumps

Classification of Bump Events

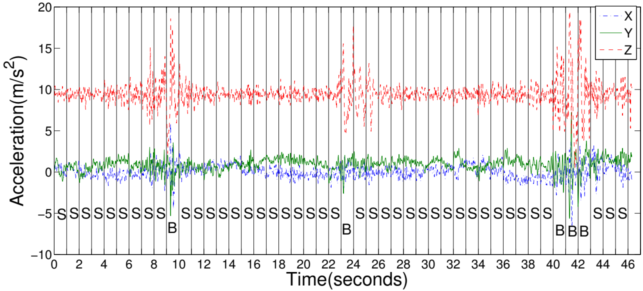


Figure: Accelerometer Data for three speedbreakers

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

Training for Braking Events

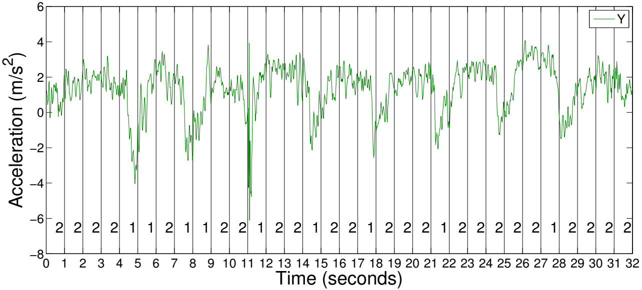


Figure: Y axis Accelerometer for braking events(with labels)

Training for Braking Events

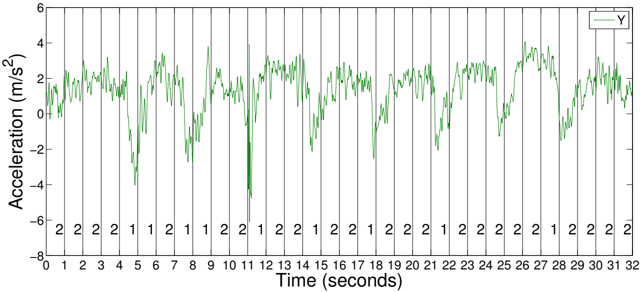


Figure: Y axis Accelerometer for braking events(with labels)

We choose δ_Y as the only feature for detecting bumps

Classification of Braking Events

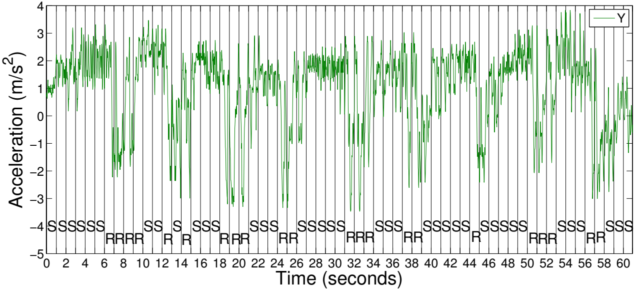


Figure: Braking events with generated class labels

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

Experimental Setup

- Used **HTC Wildfire S** and **Samsung Nexus S**
- 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

Bump Detection on Scooter: Training

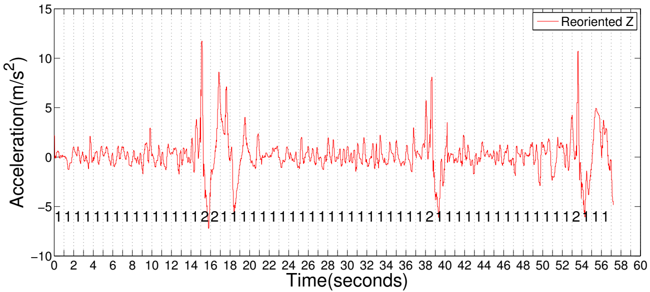


Figure: Accelerometer readings in reoriented Z-direction for scooter training data, with clusters

Bump Detection on Scooter: Testing

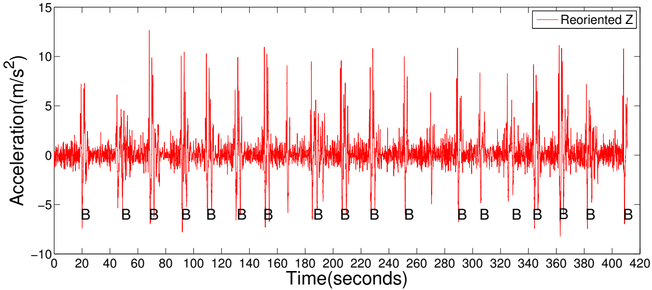


Figure: Accelerometer readings in reoriented Z-direction for scooter test data, with labels

Correctly identified 18 out of 20 bump events

Bump Detection on Auto Rickshaw: Training

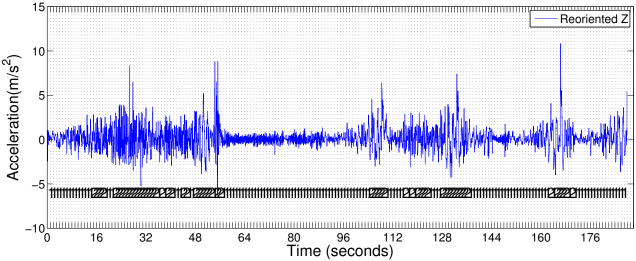


Figure: Accelerometer readings in reoriented Z-direction for autorickshaw training data, with clusters

Bump Detection on Auto Rickshaw: Testing

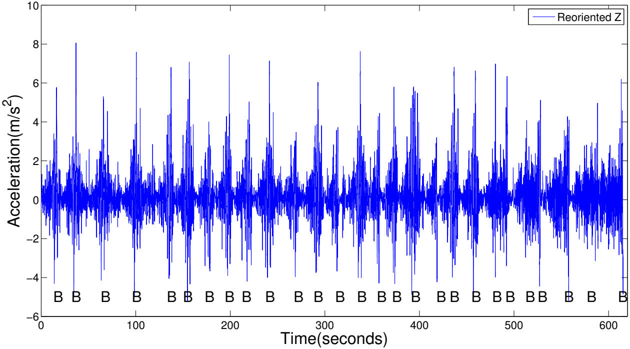


Figure: Accelerometer readings in reoriented Z-direction for autorickshaw test data, with labels

Bump Detection Experimental Results

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

Braking Detection on Scooter: Training

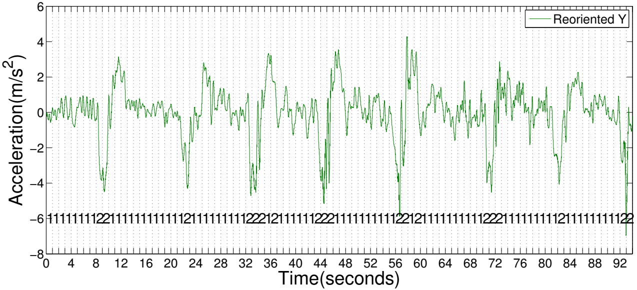


Figure: Accelerometer readings in reoriented Y-direction for scooter training data, with clusters

Braking Detection on Scooter: Testing

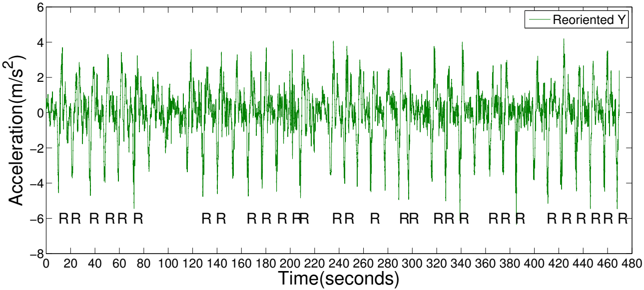


Figure: Accelerometer readings in reoriented Y-direction for scooter test data, with labels

Correctly identified 29 out of 37 braking events, with one False Positive at the 200th second

Braking Detection on Auto Rickshaw: Training

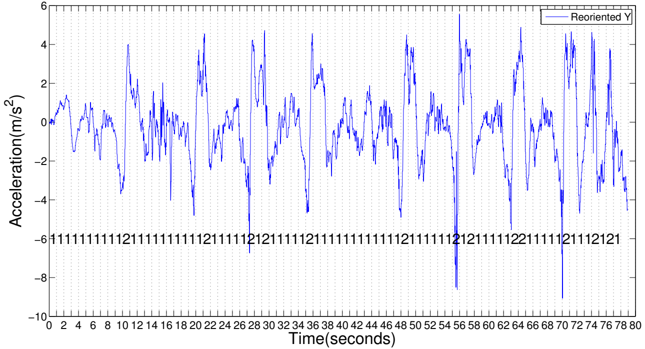


Figure: Accelerometer readings in reoriented Y-direction for autorickshaw training data, with clusters

Braking Detection on Auto Rickshaw: Testing

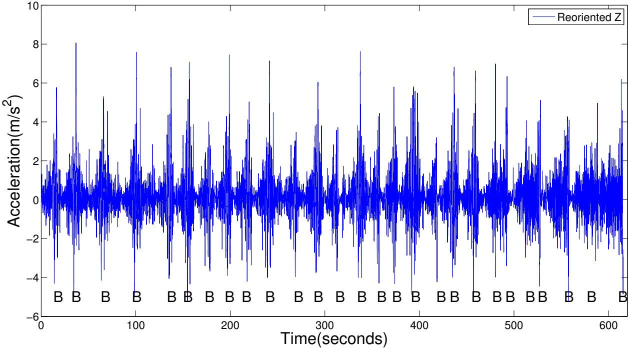


Figure: Accelerometer readings in reoriented Y-direction for autorickshaw test data, with labels

Braking Detection Experimental Results

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

Concluding Remarks on Machine Learning

- Very low false positives, low false negatives
- Single pass algorithm, low memory requirements – 16 bytes per second of training data
- σ is more robust than δ, but can't detect events of small duration
- In case of noisy data, filtering may have to be applied to use δ

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 [3] [7] [4]

Energy Savings compared to Nericell

- Per reorientation event

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

- Total Energy Saved

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

Conclusions and Future Work

- Conclusions
  - Considering the use of magnetometer reduces the energy in reorientation of accelerometer axes
  - Traffic state estimation is possible, by braking detection
  - Road state estimation is poosible, by bump detection
  - 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 state estimation is possible, by braking detection
  - Road state estimation is poosible, by bump detection
  - Scalable system, as any user having smartphone can participate
- Future Work
  - Fully implement application that can be installed by the users in their smartphones
  - Process information to annotate maps
  - Record energy consumption for a better energy model
  - Localization in energy efficient manner
  - Differentiating classes of vehicles

References I

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.

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.

References II

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.

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.

References III

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.

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?