Nericell: Rich monitoring of road and traffic conditions using mobile smartphones

Conference Paper · November 2008

DOI: 10.1145/1460412.1460444 · Source: DBLP

3 authors, including:

Prashanth Mohan
University of California, Berkeley
17 PUBLICATIONS **3,023** CITATIONS

SEE PROFILE

TrafficSense: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones

Prashanth Mohan, Venkata N. Padmanabhan, and Ramachandran Ramjee
{prmohan,padmanab,ramjee}@microsoft.com
Microsoft Research India, Bangalore

April 2008

Technical Report
MSR-TR-2008-59

We consider the problem of monitoring road and traffic conditions in a city. Prior work in this area has required the deployment of dedicated sensors on vehicles and/or on the roadside, or the tracking of mobile phones by service providers. Furthermore, prior work has largely focused on the developed world, with its relatively simple traffic flow patterns. In fact, traffic flow in cities of the developing regions, which comprise much of the world, tends to be much more complex owing to varied road conditions (e.g., potholed roads), chaotic traffic (e.g., a lot of braking and honking), and a heterogeneous mix of vehicles (2-wheelers, 3-wheelers, cars, buses, etc.). To monitor road and traffic conditions in such a setting, we present TrafficSense, a system that performs rich sensing by piggybacking on smartphones that users carry around with them. In this paper, we focus specifically on the sensing component, which uses the accelerometer, microphone, GSM radio, and/or GPS sensors in these phones to detect potholes, bumps, braking, and honking. TrafficSense addresses several challenges including virtually reorienting the accelerometer on a phone that is at an arbitrary orientation, and performing honk detection and localization in an energy efficient manner. We also touch upon the idea of triggered sensing, where dissimilar sensors are used in tandem to conserve energy. We evaluate the effectiveness of the sensing functions in TrafficSense based on experiments conducted on the roads of Bangalore, with promising results.

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://www.research.microsoft.com

1 Introduction

Roads and vehicular traffic are a key part of the day-to-day lives of people. Therefore, monitoring their conditions has received a significant amount of attention. Prior work in this area has primarily focused on the developed world, where good roads and orderly traffic mean that the traffic conditions on a stretch of road can largely be characterized by the volume and speed of traffic flowing through it. To monitor this information, intelligent transportation systems (ITS) [7] have been developed. Many of these involve deploying dedicated sensors in vehicles (e.g., GPS-based tracking units [9]) and/or on roads (inductive loop vehicle detectors, traffic cameras, doppler radar, etc.), which can be an expensive proposition and so is typically restricted to the busiest stretches of road. See [3] for a good overview of traffic surveillance technologies and Section 2 for related work.

In contrast, road and traffic conditions in the developing world tend to be more varied because of various socio-economic reasons. Road quality tends to be variable, with bumpy roads and potholes being commonplace even in the heart of cities. The flow of traffic can be chaotic, with little or no adherence to right of way protocols at some intersections and liberal use of honking.

Monitoring such varied road and traffic conditions is challenging but it holds the promise of enabling new and useful functionality. For instance, information gathered via rich sensing could be used to annotate a map, thereby allowing a user to search for driving directions that would minimize stress by avoiding chaotic roads and intersections.

To address this challenge, we present TrafficSense, a system for rich monitoring of road and traffic conditions that piggybacks on mobile smartphones. TrafficSense orchestrates the smartphones to perform sensing and report data back to a server for aggregation. Indeed, smartphones include a range of sensing and communication capabilities, in addition to computing. A phone might include any or all of a microphone, camera, GPS, and accelerometer, each of which could be used for traffic sensing functions. In addition, the phone would include a cellular radio (e.g., GSM), possibly with data communication capabilities (e.g., GPRS or UMTS).

A mobile phone-based approach to traffic monitoring is a good match for developing regions because it avoids the need for expensive and specialized traffic monitoring infrastructure. Moreover, it takes advantage of the booming growth of mobile telephony in such regions. For example, as of late 2007, India had 209 million mobile telephony subscribers, growing at an estimated 7 million each month [12]. Although the majority of users have basic mobile phones today, a large number of them, in fact more than the number of PC Internet users in India, access the internet on their phones [8], suggesting that the prospects of greater penetration of more capable phones are good. There are similar growth trends in many other parts of the world, with the total number of mobile users worldwide estimated at 2.3 billion [13]. Note that despite being mobile phone-based, our approach to traffic monitoring is distinct from prior work based on remote tracking of mobile phones by cellular operators [1, 2, 34]. Our sensing and inferences goes beyond just monitoring location and speed information, hence requiring a presence on the phones themselves.

Our focus in this paper is on the sensing component of TrafficSense; we defer a discussion of the larger system, including the aggregation server, to future work. Several technical challenges arise from our design choice to perform rich sensing and base the system on mobile smartphones. TrafficSense leverages sensors besides GPS — accelerometer and microphone, in particular — to glean rich information, e.g., the quality of the road or the noisiness of traffic. The use of an accelerometer introduces the challenge of *virtually reorienting* it to compensate for the arbitrary orientation of the phone that it is embedded in. Once the accelerometer is virtually reoriented, we need to design efficient and robust bump, brake and honk detectors in order to infer road and traffic conditions.

Moreover, since a smartphone is battery powered and is primarily someone's phone, energy-efficiency

is a key consideration in TrafficSense. To this end, we employ the concept of *triggered sensing*, wherein a sensor that is relatively inexpensive from an energy viewpoint (e.g., cellular radio or accelerometer) is used to trigger the operation of a more expensive sensor (e.g., GPS or microphone). For efficiency in communication and energy usage, each node processes the sensed data locally before shipping the processed data back to the server.

The main contributions of this work are: (1) algorithms to virtually reorient a disoriented accelerometer along a canonical set of axes and then use simple threshold-based heuristics to detect bumps and potholes, and braking (Section 5); (2) heuristics to identify honking by using audio samples sensed via the microphone (Section 6); (3) evaluation of the use of cellular tower information in dense deployments in developing countries to perform energy-efficient localization (Section 7); and (4) triggered sensing techniques, wherein a low-energy sensor is used to trigger the operation of a high-energy sensor (Section 8). Finally, we have implemented most of these techniques on smartphones running Windows Mobile 5.0 (Section 9).

There are two important issues that we do not address in this paper. One is the question of *privacy* of the users whose phones participate in TrafficSense. It may be possible to achieve good enough privacy simply by suppressing the identity of a participating phone (e.g., its phone number) when reporting and aggregating the sensed data. A more sophisticated privacy-aware community sensing approach that incorporates formal models of sharing preferences is presented in [28]. The second issue is of providing incentives for participation in TrafficSense. Providing incentives for participation in such decentralized systems is an active area of research (e.g., [15]) and we may be able to leverage this for TrafficSense.

2 Related work

Intelligent transportation systems [7] have been proposed and built to leverage computing and communication technology for various purposes: traffic management, routing planning, safety of vehicles and roadways, emergency services, etc. We focus here on work that is most relevant to TrafficSense.

There has been much work on systems for traffic monitoring, both in the research world and in the commercial space. Many of these systems leverage vehicle-based GPS units (e.g., as in GM's OnStar [9]) that track the movement of vehicles and report this information back to a server for aggregation and analysis. For instance, CarTel [25] includes a special box installed in vehicles to monitor their movements using GPS and report it back using opportunistic communication across a range of radios (WiFi, Bluetooth, cellular). This information is then used for applications such as route planning.

Recent work on Surface Street Traffic Estimation [37] also uses GPS-derived location traces but goes beyond just estimating speed to identifying anomalous traffic situations using both the temporal and spatial distributions of speed. For instance, the authors are able to distinguish between traffic congestion and vehicles halting at a traffic signal.

Operational services, both commercial and otherwise, have been built using GPS information as well as information from other traffic sensors deployed in an area (inductive loop vehicle detectors, traffic cameras, doppler radar, etc.). Examples include the Washington State SmartTrek system in the U.S. [20] (which includes, among other things, the Busview service [10], to track the city buses in the Seattle area) and the INRIX system [6] for predicting traffic based on historical data.

There has also been work on leveraging mobile phones carried by users as traffic probes. Smith *et al.* [34] report on a trial conducted in Virginia in 2000, which was based on localizing mobile phones using information gathered at the cellular towers. This study made a number of interesting observations, including that the sample density (at the place and time of this study) was only sufficient for estimating speed with moderate accuracy (within 10 mph = 16 kmph), and that there was a tendency to underestimate speed

because of samples from stationary phones located near the roadway. Regardless, the rapid growth in mobile phone penetration has spurred the deployment of similar systems in other locations worldwide, including in Bangalore [2].

Much of the work on ITS has used GPS-based localization and some of it has used localization performed at cell towers. However, there has been separate work on enabling a mobile phone to locate itself, whether based on GSM signals [18, 35], with a median accuracy under 100m in the outdoors measured in the Seattle area, or based on WiFi [19], with a median accuracy of 13-40m in an area with dense WiFi coverage. It is possible to use these localization techniques in the context of an ITS, either to complement GPS or as an alternative.

Other forms of sensing, besides localization, have also been employed in ITS systems. Accelerometers are used for automotive safety applications such as detecting crashes to deploy airbags and to possibly also notify emergency services automatically (e.g., Veridian [26]). Accelerometers and strain gauges coupled with cameras have been used for structural monitoring of the transportation infrastructure [17]. In recent work, the Pothole Patrol (P^2) system [22] performs road surface monitoring by using special-purpose devices with 3-axis accelerometers and GPS sensors mounted on the dashboard of cars. It tackles the challenging problem of not only identifying potholes but also differentiating potholes from other road anomalies. The use of a special-purpose device mounted in a known orientation, which simplifies the analysis, is a key distinction of P^2 compared to our work on TrafficSense, which leverages smartphones that users happen to carry with them.

To put it in context, our work on TrafficSense builds on prior work but is distinct from it in several ways. We do not replicate the significant body of prior work on estimating the speed of traffic flow [34, 37] and driving patterns [29] based on location traces, and presenting this information to users in an appropriate form [30]. Instead, TrafficSense focuses on novel aspects of sensing varied road and traffic conditions, such as bumpy roads and noisy traffic. Furthermore, TrafficSense uses smartphones that users happen to carry with them, depending only on capabilities that are already available in some phones and that are likely to be available in many more in the coming years. By piggybacking on an existing platform, TrafficSense avoids the need for specialized and potentially expensive monitoring equipment to be installed, whether on vehicles as in [22, 25] or as part of the infrastructure as in SmartTrek [20]. But building on top of a mobile phone platform introduces challenges, for instance, with regard to accelerometer orientation, energy efficiency and device localization, which we address in TrafficSense. Finally, TrafficSense falls under the active area of research called Participatory Sensing [14] and can leverage solutions to common challenges in this area such as data credibility and privacy.

3 Experimental Setup

We discuss the hardware and software setup used for our work and the measurement data that we gathered.

3.1 Smartphone Capabilities

A smartphone may include any or all of the following capabilities of relevance to TrafficSense:

- **Computing:** CPU, operating system, and storage that provides a programmable computing platform.
- **Communication:**
  - *Cellular:* a radio for basic cellular voice communication (e.g., GSM), available in all phones.

  - *Cellular data:* e.g., GPRS, EDGE, UMTS, provided by the cellular radio.
  - *Local-area wireless:* radios for local-area wireless communication (e.g., Bluetooth, WiFi).

- **Sensing:**
  - *Audio:* a microphone.
  - *Localization:* a GPS receiver.
  - *Motion:* an accelerometer, sometimes included for functions such as gesture recognition.
  - *Visual:* a camera, although TrafficSense does not make use of this at present.

These capabilities are not only within the realm of engineering possibility, but in fact there exist smartphones on the market that include most or all of the above capabilities in a single package. For example, the Nokia N95 includes all whereas the HP iPAQ hw6965 includes all except an accelerometer and the Apple iPhone includes all except GPS. We emphasize, however, that TrafficSense does not require all participating phones to include each of these capabilities.

3.2 Hardware and Software Setup

Despite the availability of such capable smartphones, our experimental setup is complicated a little by hardware and software constraints of the devices available to us. We describe here the devices that we use.

- *HP iPAQ hw6965 [5]:* This Pocket PC running Windows Mobile 5.0 includes a GSM/GPRS/EDGE radio, Bluetooth 1.2, 802.11b, and a Global Locate GPS receiver.
- *HTC Typhoon:* For cellular localization, we use rebranded HTC Typhoon phones, specifically the Audiovox SMT5600, iMate SP3, and Krome iQ700, which feature a tri-band GSM radio and run Windows Mobile 2003. As noted in [18, 32] the HTC Typhoon is convenient to use for this purpose because it makes available information about multiple cell towers in the vicinity. All phones (including our HP iPAQs) have this tower information (which is needed to perform handoffs) but do not expose it to user-level software, although there is no fundamental reason why they could not.
- *Sparkfun WiTilt accelerometer [11]:* The Sparkfun WiTilt combines a Freescale MMA7260Q [4] 3-axis accelerometer sensor with a Bluetooth radio to enable remote reading. The Freescale MMA7260Q accelerometer sensor has a selectable sensitivity ranging from +/-1.5g to +/-6g and a sampling frequency of up to 610 Hz.

While the HP iPAQ is the centerpiece of our setup, we use the WiTilt as the accelerometer sensor and use the Typhoon for cellular localization.

3.3 Trace Collection

Much of our experimental work was set in Bangalore. We gathered GPS-tagged cellular tower measurements during several drives over the course of 4 weeks. Separately, we gathered GPS-tagged accelerometer data measurements on drives on some of the same routes over the course of 6 days. [1] We also gathered cellular

[1]The only reason that the accelerometer measurements were made separately from the cellular measurements was that we procured the accelerometers later.

Location	Dates	Duration (hours)	Dist. (km)	Information recorded
Bangalore	09/03/07-09/19/07 11/20/07-11/22/07	19.5	377	GPS, GSM
Seattle	09/25/07-09/28/07	4.1	183	GPS, GSM
Bangalore	03/30/08 04/08/08-04/13/08	4.0	62	GPS, accel.

Table 1: **Summary of data gathered on drives**

tower measurements over the course of a few days in the Seattle area. Table 1 summarizes all of the data that we gathered on drives through traffic. Figure 1 shows a map of Bangalore with the drive routes highlighted.

In addition to drive data, we gathered accelerometer data over specific sections of road (selected for their bumps, potholes, etc.) at controlled speeds and using different kinds of vehicles (2-wheelers and 4-wheelers). We also recorded the sound of several vehicles honking. Figure 2 shows a typical chaotic intersection with different vehicle types, each approaching the intersection from different orientations with no adherence to right of way protocols. These intersections are typically also characterized by significant amount of braking and honking.

4 Overview

The richness of sensing that TrafficSense encompasses is motivated by the wide applicability we envisage for the system. The system could be used to annotate traditional traffic maps with information such as the bumpiness, noisiness, and level of traffic chaos, for the benefit of the traffic police, the road works department, and ordinary users. For instance, a user might search for a route that minimizes the number of chaotic intersections to be traversed, thereby optimizing for "blood pressure" rather than for distance or time. While these applications serve as the motivation, our focus in this paper is on the sensing component of TrafficSense, specifically, how to efficiently use the accelerometer, microphone, GSM and GPS sensors in mobile smartphones to detect bumps and potholes, braking, and honking, and to determine location in an energy-efficient manner.

In Section 5, we focus on sensing using a 3-axis accelerometer. Given that a mobile smartphone and its embedded accelerometer could be in any arbitrary orientation, we first discuss our algorithm for *virtually reorienting* a disoriented accelerometer automatically. Using real drive data from reoriented as well as well-oriented accelerometers, we evaluate the efficacy of our reorientation algorithm as well as our simple heuristics to detect bumps and potholes, and braking. In Section 6, we describe how audio sensed using the microphone can be used to detect honks. In Section 7, we discuss and evaluate the accuracy of energy-efficient coarse-grain localization and traffic speed estimation using GSM cellular tower information.

Given the energy costs of the different sensors, as indicated in Table 2, TrafficSense only keeps its GSM radio (which has to be turned on anyway for the phone to function) and the accelerometer on continually. It uses input from these two devices to *trigger* the turning on of the other sensors, for instance, to obtain a precise location fix using GPS. We discuss this and other examples of triggered sensing in Section 8.

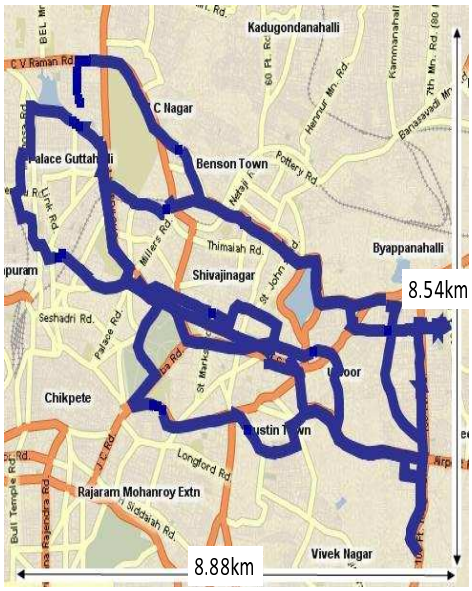

Figure 1: **Map of Bangalore with drive routes highlighted**

Figure 2: **A typical chaotic road intersection with variety of vehicles at loggerheads**

5 Acceleration

In this section, we discuss how TrafficSense uses the accelerometer on a phone to sense road and traffic conditions.

5.1 Framework

As noted in Section 3.2, we use a 3-axis accelerometer for our work. The accelerometer has a 3-dimensional Cartesian frame of reference with respect to itself, represented by the orthogonal x, y, and z axes. In addition, we define a Cartesian frame of reference with respect to the vehicle that the accelerometer (or rather the phone that it is part of) is in. The vehicle's frame of reference is represented by the orthogonal X, Y, and Z axes, with X pointing directly to the front, Y to the right, and Z into the ground. See Figure 3 for an illustration.

Note the distinction between the lower case letters (x, y, z) used to represent the accelerometer's frame of reference and the upper case letters (X, Y, Z) used to represent the vehicle's frame of reference. If (x, y, z) is aligned with (X, Y, Z), we say that the accelerometer is *well-oriented*. Otherwise, we say that it is *disoriented*.

The accelerometer readings along the 3 axes are denoted by a_x, a_y, and a_z. If the accelerometer is well-oriented, these readings would also correspond to a_X, a_Y, and a_Z along the vehicle's axes. All acceleration values are expressed in terms of g, the acceleration due to gravity (9.8 m/s^2). Also, we set the sampling

Mode	**Life Time** Includes Phone Idle	**Power (mW)** For given mode only
Phone Idle	24h 18m	182.7
Bluetooth (BT) Idle	22h 13m	17.1
BT Device Inquiry	10h 46m	229.5
BT Service Discovery	7h 53m	380.0
WiFi Idle	4h 39m	771.8
WiFi Beacon (Sending)	4h 36m	782.0
WiFi Scan (Receiving)	2h 59m	1298.8
GPS	5h 32m	617.3
Microphone	10h 54m	223.2
Accelerometer (per spec.)	24h 5m	1.65
Accel. with Bluetooth	19h 56m	40

Table 2: **Power usage for various activities**

frequency of the accelerometer to 310 Hz, unless noted otherwise.

Finally, we note that ours is a DC accelerometer, which means that it is capable of measuring "static" acceleration. For instance, even when a well-oriented accelerometer is stationary, it reports $a_z = 1g$. In essence, the measurement reported by the accelerometer is a function of the *force* exerted on its sensor mechanism, not the textbook definition of acceleration as the rate of change of velocity. For the same reason, when the vehicle accelerates (which would represent a positive acceleration along X according to the textbook definition), our accelerometer would experience a force pressing it backwards and hence report a negative acceleration along X.

5.2 Determining Accelerometer Orientation

In general, the phone (or, rather, the accelerometer embedded in it) and its (x,y,z) axes could be in an arbitrary orientation with respect to the vehicle and its (X,Y,Z) axes. Furthermore, this orientation could change over time as the phone is moved around. A phone that is *disoriented* in this manner makes it non-trivial for its accelerometer measurements to be used to infer road and traffic conditions. For instance, if z were aligned with X (i.e., it points to the front rather than down), episodes of sharp acceleration and deceleration (i.e., horizontal acceleration) might be mistaken for bumps on the road (i.e., vertical acceleration). Thus, before the accelerometer measurements can be used, it is important for us to *virtually reorient* the accelerometer to compensate for its disorientation. The need to address this challenge is a key distinction of our approach based on phones from the prior work on Pothole Patrol [22] that leverages a dedicated accelerometer mounted at a known orientation.

We define the canonical orientation of (x,y,z) to be the one that corresponds to (X,Y,Z). In general, any arbitrary orientation of (x,y,z) can be arrived at by applying rotations about X, Y, and Z in sequence, starting with the canonical orientation. Our goal is to infer the angles of rotation about each axis. While we could work with such a framework, it yields multi-factor trigonometric equations that are complex to solve.

5.2.1 Euler Angles

An alternative but equivalent framework is based on Euler angles [23, 36], which, as it turns out, simplifies our calculations significantly. There are a number of formulations of Euler angles, but we only describe the formulation (termed $Z-Y-Z$) that we use in our work. Any orientation of the accelerometer can be represented by a *pre-rotation* of ϕ_{pre} about Z, followed by a *tilt* of θ_{tilt} about Y, and then a *post-rotation* of ψ_{post} again about Z. All angles are measured counter-clockwise about the corresponding axis. (The subscripts to ϕ_{pre}, θ_{tilt}, and ψ_{post} are not needed, but we include these for clarity.)

That these three Euler angles are sufficient to represent any orientation might seem counter-intuitive since there is no rotation about X and thus one might erroneously conclude that there is no impact on y. However, because of the pre-rotation, the tilt would in fact affect both x and y in general. For example, if the pre-rotation were by 90^o, y would be in line with X, so the tilt about Y would in fact impact y.

5.2.2 Estimating Pre-rotation and Tilt

With this framework in place, we now describe our procedure for determining the orientation of the disoriented accelerometer. When the accelerometer is stationary or in steady motion, the only acceleration it will experience is that due to gravity, along Z. (Recall that our accelerometers report the strength of the force field, so $a_Z = 1g$ despite the accelerometer being stationary.) The tilt operation is the only one that changes the orientation of z with respect to Z. So $a_z = a_Z cos(\theta_{tilt})$. Since $a_Z = 1$, we have

$$\theta_{tilt} = cos^{-1}(a_z) \tag{1}$$

Pre-rotation followed by tilt would also result in non-zero a_x and a_y due to the effect of gravity. As explained in Appendix A, we have:

$$\phi_{pre} = tan^{-1}(\frac{a_y}{a_x}) \tag{2}$$

To estimate θ_{tilt} and ϕ_{pre} using Equations 1 and 2, we could identify periods when the phone is stationary (e.g., at a traffic light) or in steady motion, say using GPS to estimate speed. However, a simpler approach that we have found works well in practice is to just use the *median* values of a_x, a_y, and a_z over a 10-second window. The median value over a window of this length turns out to be remarkably stable, even during a bumpy drive. Intuitively, any spike in acceleration would tend to be momentary and would settle back around the median within a few seconds or less. For example, if the vehicle gets lifted up by a bump, thereby causing a spike in the g-force, it would descend back soon enough, causing a spike in the reverse direction; the median itself remains largely unaffected.

Thus by computing a_x, a_y, and a_z over short time windows, we are able to estimate θ_{tilt} and ϕ_{pre} on an ongoing basis. Any significant change in θ_{tilt} and ϕ_{pre} would indicate a significant change in the phone's orientation. However, the converse is not always true. If the phone were carefully rotated about Z, θ_{tilt} and ϕ_{pre} would remain unaffected.

5.2.3 Estimating Post-rotation

Since post-rotation (like pre-rotation) is about Z, it has no impact with respect to the gravitational force field, which runs parallel to Z. So we need a different force field with a known orientation that is not parallel to Z, to be able to estimate the angle of post-rotation, ψ_{post}. In practice, we would need this force field to have a *significant* component in a known direction orthogonal to Z.

The acceleration and braking of a vehicle travelling in a straight line produces a force field in a known direction, X, in line with the direction of motion of the vehicle. However, these would have little impact on

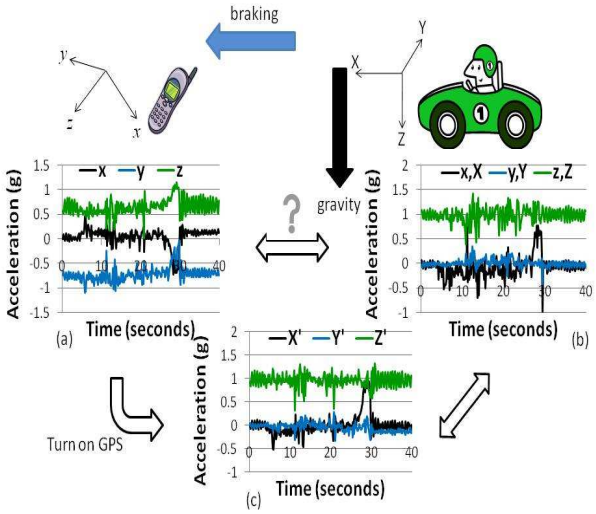

Figure 3: **This figure illustrates the process of virtually reorienting a disoriented accelerometer. (a) shows the measurements made at a disoriented accelerometer, which unsurprisingly do not match with the measurements shown in (b), corresponding to a well-oriented accelerometer. However, after the disoriented accelerometer has been virtually reoriented, its corrected measurements shown in (c) match those in (b) quite well.**

the orthogonal directions, in particular, on Y. In general, braking tends to be sharper than acceleration, so we focus our attention on braking. For example, if a car travelling at 50 kmph $\approx$ 31 mph brakes to a halt in 10 seconds, a_X would be be about $0.14g$, a small but still noticeable level.

We use GPS to monitor the vehicle's speed and thereby identify periods of sharp deceleration without a significant curve in the path (i.e., the GPS track is roughly linear). During such a period, we know that there will be a noticeable, even if transient, force field along X, with little impact on Y. Given the measured accelerations (a_x, a_y, a_z), and the angles of pre-rotation (ϕ_{pre}) and tilt (θ_{tilt}), we estimate the angle of post-rotation (ψ_{post}) as the one that *maximizes* our estimate, $a_X^{'}$, of the acceleration along X, which is the direction of braking. Note that $a_X^{'}$ is an estimate and hence is not necessarily equal to the true a_X.

As explained in Appendix B, this maximization procedure yields:

$$\psi_{post} = tan^{-1} \left(\frac{-a_x sin(\phi_{pre}) + a_y cos(\phi_{pre})}{(a_x cos(\phi_{pre}) + a_y sin(\phi_{pre}))cos(\theta_{tilt}) - a_z sin(\theta_{tilt})} \right) \quad (3)$$

To estimate ψ_{post}, we first estimate ϕ_{pre} and θ_{tilt} using Equations 1 and 2. We then identify an instance of sharp deceleration using GPS data and record the *mean* a_x, a_y, and a_z during this period. In our experiments, we use just the first few seconds (typically 2 seconds) of the deceleration to compensate for the time lag in the speed estimate obtained from GPS. Note that we use the mean here, unlike the median used in Section 5.2.2, because we want to record the transient surge because of the sharp deceleration.

Compared to estimating ϕ_{pre} and θ_{tilt}, estimating ψ_{post} is more elaborate and expensive, requiring GPS to be turned on. So we monitor ϕ_{pre} and θ_{tilt} on an ongoing basis, and only if there is a significant change

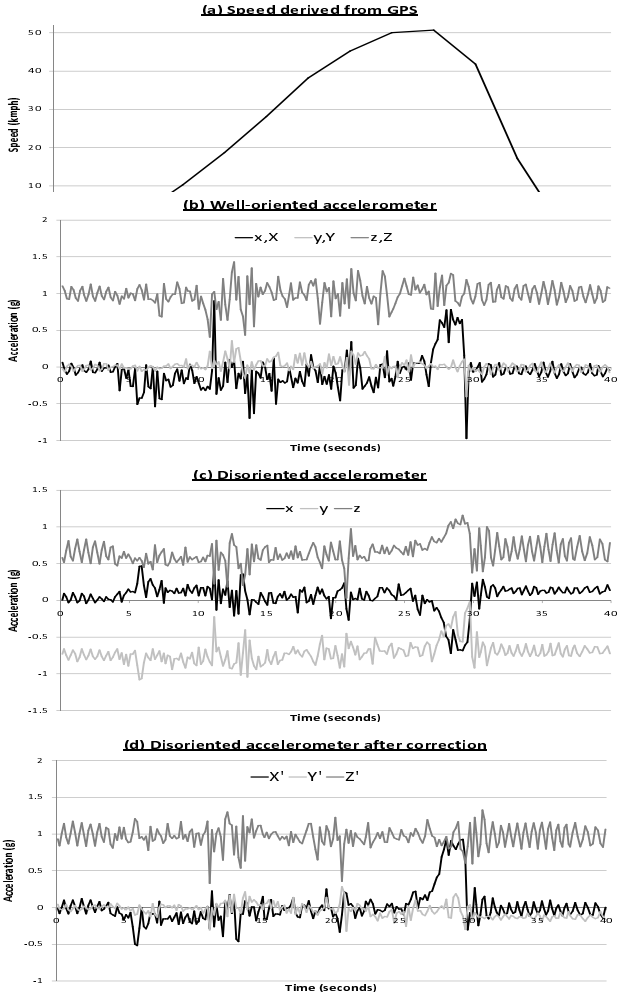

Figure 4: **Estimating the orientation of a disoriented accelerometer based on sharp deceleration starting at around 27 seconds:** $\phi_{pre} = -88^o$**,** $\theta_{tilt} = 49^o$**,** $\psi_{post} = -135^o$

in these or there is other evidence that the phone's orientation may have changed (Section 5.3), do we turn on GPS and run through the process of estimating ψ_{post} afresh.

5.2.4 Validating Virtual Reorientation

To validate the virtual reorientation algorithm described above, we conduct an experiment that involves measurements made with three accelerometers: ACL-1 and ACL-2, which are well-oriented, and ACL-3, which is disoriented. We make these measurements during a drive, which includes episodes of acceleration and braking, and use these to estimate ϕ_{pre}, θ_{tilt}, and ψ_{post} for ACL-3, using Equations 1, 2, and 3, respectively. Using these estimates and the measured a_x, a_y, and a_z for ACL-3, we estimate $a_X^{'}$, $a_Y^{'}$, and $a_Z^{'}$. We then compare these estimates with the measured values of a_X, a_Y, and a_Z from the well-oriented accelerometers, ACL-1 and ACL-2. A good match would suggest that the estimates of ϕ_{pre}, θ_{tilt}, and ψ_{post} are accurate.

As an illustration, Figure 4 shows to a sharp deceleration during the period of 27-30 seconds, which

	W_1-W_2	$\phi_{pre}/\theta_{tilt}/\psi_{post}$	D_3-W_1	R_3-W_1	D_3-W_2	R_3-W_2
1	0.90	$7^o/38^o/106^o$	0.30	0.88	0.20	0.91
2	0.75	$174^o/34^o$/-107^o	0.43	0.72	0.54	0.87
3	0.94	$174^o/34^o$/-107^o	0.59	0.84	0.67	0.90
4	0.74	$4^o/42^o/12^o$	0.65	0.72	0.63	0.68
5	0.76	$3^o/44^o$/-1^o	0.62	0.71	0.69	0.79
6	0.78	-$80^o/42^o/121^o$	0.65	0.73	0.64	0.73

Table 3: **Cross-correlation between the X components of the disoriented accelerometer (ACL-3) and each of the well-oriented accelerometers (ACL-1 and ACL-2) across several episodes of acceleration and braking. We report the cross-correlation numbers for before (D-W) and after (R-W) virtual reorientation is performed, along with the reorientation angles. We also report the W-W cross-correlation numbers between the two well-oriented accelerometers, to provide a point of comparison.**

generates a surge in a_X alone, as recorded by ACL-1, one of the two well-oriented accelerometers (Figure 4(b)). The disoriented accelerometer, ACL-3, on the other hand, records surges or dips in its a_x, a_y, and a_z (Figure 4(c)). However, after ACL-3's orientation is estimated and compensated for, $a^{'}_X$, $a^{'}_Y$, and $a^{'}_Z$ are estimated (Figure 4(d)). Figure 4(d) only shows a surge in $a^{'}_X$, which is consistent with Figure 4(b).

To quantify the effectiveness of virtual reorientation, we compute the cross-correlation between the measurements from the reoriented accelerometer and those from the well-oriented accelerometers. The cross-correlation between two time series, $x[i]$ and $y[i]$, is defined as:

$$r = \frac{\sum_{i=0}^{N} \{(x[i] - \bar{x}) * (y[i] - \bar{y})\}}{\sqrt{\sum_{i=0}^{N} (x[i] - \bar{x})^2} * \sqrt{\sum_{i=0}^{N} (y[i] - \bar{y})^2}}$$

$\bar{x}$ and $\bar{y}$ correspond to the means of the two time series. When the two time series are perfectly correlated, $r = 1$. When they are entirely uncorrelated, $r = 0$.

As is clear from Figure 4, measurements from an accelerometer are noisy, presumably because of artifacts of its analog sensor mechanism. This has a two implications for our evaluation. First, the cross-correlation between measurements from two accelerometers is low during periods when the measurements only comprise (uncorrelated) noise, with little or no signal. However, since our interest is in periods when there is a signal, say due to braking or a bump, we only consider the measurements during such periods of activity in our computation of cross-correlation. In the experiment we conducted, this would correspond to the X component during the episodes of acceleration and braking. Second, the presence of noise even during such periods of interest means that two accelerometers may not agree perfectly, even if they are both well-oriented. So, in addition to reporting the cross-correlation between the virtually reoriented accelerometer (ACL-3) and each of the well-oriented ones (ACL-1 and ACL-2), we also report the cross-correlation between the two well-oriented accelerometers (ACL-1 and ACL-2) to provide a point of comparison.

Table 3 shows the cross-correlation numbers between each pair of accelerometers across several episodes of acceleration and braking. Each episode lasted 15-20 seconds and the accelerometer sampling frequency was set to 25 Hz, yielding a time series comprising 375-450 samples. The accelerometers were placed on the rear seat of a Toyota Qualis vehicle. The orientation of ACL-3 was held fixed across certain pairs of episodes and was changed between others. We observe that, in general, the the cross-correlation improves significantly when we go from having a disoriented accelerometer to one that has been virtually reoriented.

For example, in episode 1, the cross-correlation between ACL-3 and ACL-1 improves from 0.30 to 0.88 after virtual reorientation is performed. In some cases, the improvement is smaller, for example, going from 0.62 to 0.71 in episode 5. The reason for the smaller improvement in episode 5 is that the angles of pre-rotation (ϕ_{pre}) and post-rotation (ψ_{post}) are small (3^o and -1^o, respectively). So even with disorientation, braking causes a surge in a_x, albeit reduced in magnitude because of the tilt, θ_{tilt}.

We also observe that the cross-correlation numbers after virtual reorientation (the *R-W* columns) are comparable to those between the two well-oriented accelerometers (the *W-W* column). Furthermore, these cross-correlation numbers, while high, are significantly lower than the cross-correlation of 1.0 that perfect correlation would yield. The lack of perfection comes from noise spikes, which motivates our insistence in Section 5.4.3 and Section 5.4.1 on looking for sustained surges rather than momentary spikes to detect bumps and braking, unless the spikes are much larger in magnitude than those due to noise.

To conclude, our results in this section suggest that with virtual reorientation, a disoriented accelerometer has the potential to be almost as effective as a well-oriented accelerometer for the purpose of monitoring road and traffic conditions.

5.3 Detecting User Interaction

When a phone is being interacted with by the user, it would experience extraneous accelerations. Thus, we would like to neglect the accelerometer readings such periods. We use two techniques to detect user interaction. To detect orientation changes, we use the technique described in Section 5.2.2, with the caveat noted there. To detect other forms of user interaction, we look for one or more of the following: key presses on the phone's keypad, mouse movements, on-going or recently concluded phone calls.

5.4 Inferring Road and Traffic Conditions

We now present analyses of accelerometer measurements for detecting brakes and potholes in the road.

5.4.1 Braking Detection

We first consider the problem of detecting braking events. The frequency of braking on a section of road is indicative of drive quality. A high incidence of braking could be because of poor road conditions (e.g., poor lighting) that make drivers tentative or because of heavy and chaotic traffic. While GPS could be used to detect braking, doing so would incur a high energy cost, as indicated in Table 2. Furthermore, GPS-based braking detection is challenging at low speeds because of the GPS localization error of 3-6 m. So in TrafficSense, we focus on an alternative approach, namely, leveraging the accelerometer in a phone for braking detection.

In general, braking would cause a surge in a_X because the accelerometer would experience a force pushing it to the front. The surge can be significant even when the brake is applied at low speed. If a vehicle travelling at 10 kmph brakes to a halt in 1 second, that would result in an average surge of $0.28g$ in a_X and possibly much larger spikes. Figure 4(b) and (d) clearly show a sustained surge in a_X corresponding to a braking event.

The persistence of the surge over relatively long timeframes (a second or longer) makes brake detection an easier problem than detecting potholes where the signal lasts only fractions of a second (see Section 5.4.3). To detect the incidence of braking, we compute the mean of a_X over a sliding window N seconds wide. If the mean exceeds a threshold T, we declare that as a braking event.

Accelerometer (threshold T (g))	False Negative		False Positive	
	Rate	Change in speed avg(max)	Rate	Change in speed avg(min)
ACL-1 (T=0.11)	4.4%	15(16)	22.2%	12(10)
ACL-1 (T=0.12)	11.1%	16(18)	15.5%	12(9)
ACL-3 (T=0.11)	4.4%	15(16)	31.1%	12(9)
ACL-3 (T=0.12)	11.1%	16(18)	17.7%	12(9)

Table 4: **False Positives/Negatives of Brake detector**

In order to evaluate our braking detector, we need to establish the ground truth. Ideally, we would have liked to use data from the car's CAN bus for obtaining accurate information on the ground truth, but we did not have access to it. So for the evaluation presented in this section, we use GPS-based braking detection to establish the ground truth. To minimize the impact of GPS's localization error noted above, we employ a conservative approach that only considers sustained braking events that last several seconds. (See Section 5.4.2 for an evaluation of the same brake detector on sharp brakes at slow speeds with manually established ground truth.) Using a trace of GPS location estimates, say $...loc_{i-1}, loc_i, loc_{i+1}, ...$, we compute instantaneous speed at time i as $distance(loc_{i-1}, loc_{i+1})/2$. Once we have the instantaneous speed values, we define a brake as deceleration of at least $1m/s^2$ sustained over at least 4 seconds (i.e., a decrease in speed of at least 14.4 kmph over 4 seconds).

We use a 75-minute long drive over 35 km of varied traffic conditions to evaluate our braking detection heuristic. Using the GPS-derived ground truth described above, we detect a total of 45 braking events in this trace. This drive also includes data from two accelerometers, one well-oriented (ACL-1) and the other disoriented (ACL-3). The virtual reorientation procedure from Section 5.2 was applied to measurements from ACL-3, before it was fed into our braking detection heuristic. We set the parameters of the braking detection heuristic to correspond to the definition of a brake. We compute the mean of X-values over a $N = 4$ second window and depict results for threshold values of T=0.11g and T=0.12g (i.e., 10-20% more conservative than the $1m/s^2$ deceleration threshold applied on the GPS trace to establish the ground truth).

The results of applying our braking detection heuristic are shown in Table 4. The percentage of false positives/negatives and also the magnitude of the change in speed during the false positive/negative events are shown. The first observation is that the results using accelerometer ACL-1 agree quite well with the results using the reoriented accelerometer ACL-3. Thus, *the virtual reorientation algorithm preserves the characteristics of the accelerometer measurements sufficiently to allow the braking detector to work effectively.* Second, while the false positive rates seem high at 15-31%, when we examine the trace, we find that each of these *false positive events actually correspond to a deceleration event, albeit of lesser magnitude than our earlier definition of a brake.* Based on the GPS-derived ground truth, the average (minimum) speed decrease over four seconds at these events was 12 kmph (9 kmph), just short of our target of 14.4 kmph threshold. Note that a difference of 5kmph over 4 seconds corresponds to a location change of 5.6 m, which is well-within the localization error of GPS. Even if the GPS estimates were accurate, the false positives would simply correspond to less sharp brakes and are thus not problematic. In the case of false negatives, the rate is lower overall at 4-11%. The few false negatives again correspond to borderline events, with an average (maximum) speed decrease of 15 kmph (18 kmph), which when compared to the threshold of 14.4 kmph is again well within the localization error of GPS.

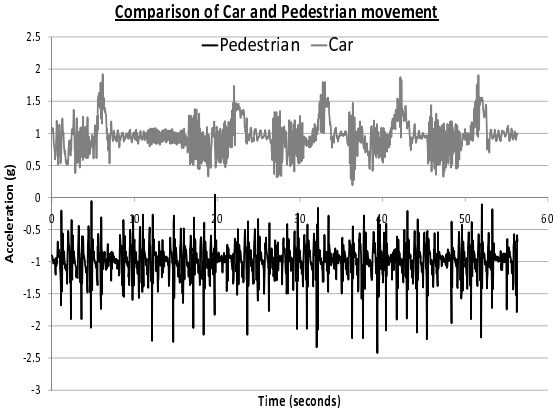

Figure 5: **X-axis accel. for a car in traffic and a pedestrian**

5.4.2 Stop-and-Go Traffic vs. Pedestrians

Given that TrafficSense piggybacks on mobile phones, we have to be able to differentiate between phones that are in vehicles stuck *in-traffic* from *out-of-traffic* phones that are in parked cars or are carried by pedestrians walking alongside the road. Otherwise, the estimation of traffic conditions might be unnecessarily pessimistic, as observed in [34].

We use accelerometer readings to differentiate pedestrian motion (and stationary phones) from vehicular motion in stop-and-go traffic. The top curve in Figure 5 shows a_X (offset by $+1g$ for clarity) in the case of a vehicle moving at 5-10 kmph in stop-and-go traffic, applying its brakes repeatedly. The braking episodes, which are annotated manually to establish the ground truth, cause surges in a_X that can be clearly seen and are also picked out by our braking detection heuristic from Section 5.4.1, with no false positives or negatives. Note that despite the low speed, and hence the surges due to the braking episodes lasting for a shorter duration than those in the drive in Section 5.4.1, the same setting of N=4 seconds works well because the surge, when averaged over a window of this duration, still exceeds the detection threshold of T=0.11g.

The bottom curve in Figure 5 shows a_X (offset by $-1g$ for clarity) for a pedestrian. The accelerometer is placed in the subject's trouser pocket, which results in larger spikes in a_X than if it were placed in a shirt pocket or in a belt clip. Despite the significant spikes in a_X associated with pedestrian motion, there is no surge that persists, unlike with the braking associated with stop-and-go traffic. When we applied our brake detection heuristic to different pedestrian accelerometer traces (with the accelerometer placed in the trouser pocket, shirt pocket, bag etc.), no false positives (i.e., brakes) were detected.

This limited evaluation suggests that our brake detection heuristic has significant potential to be used to distinguish between vehicles in stop-and-go traffic and pedestrians.

5.4.3 Bump Detection

We now consider the problem of detecting a pothole or bump on the road. A bump could arise due to a variety of reasons, both unintended (e.g., potholes), and intended (e.g., speed bumps to slow vehicles down). We do not attempt to distinguish between these causes here as we believe that an external database of the

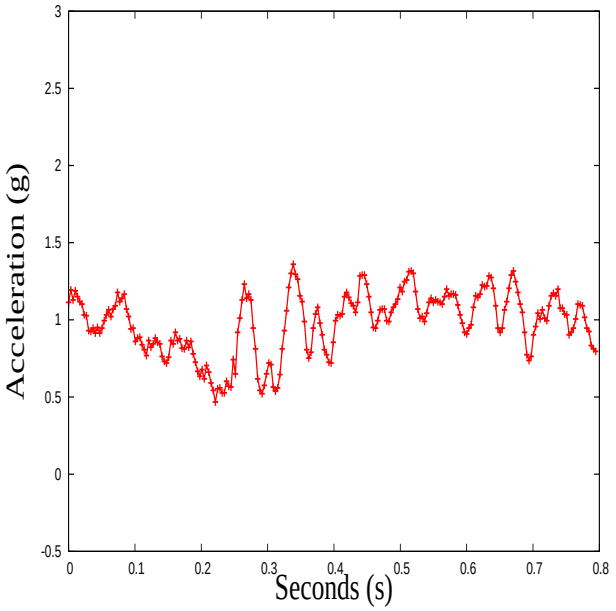

Figure 6: a_Z **when traversing a bump at low speed**

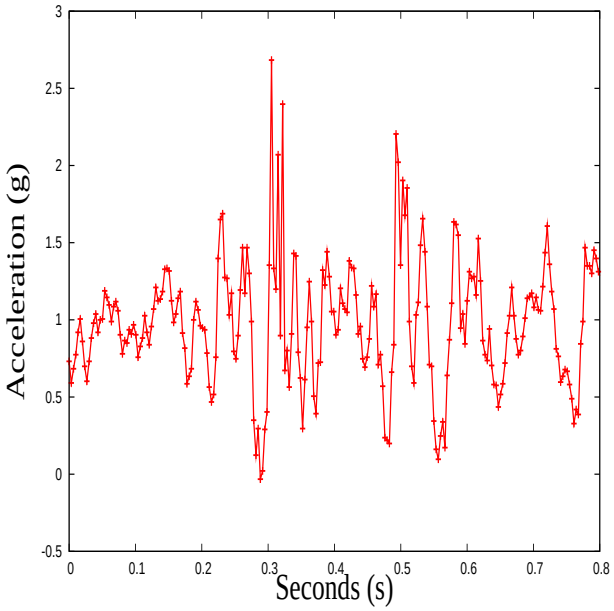

Figure 7: a_Z **when traversing a bump at high speed**

intended bumps could be used to post-process the bump events reported by TrafficSense to filter out the ones due to intended causes.

As discussed in [22], the problem of bump detection is challenging for a number of reasons. A major challenge is establishing the ground truth. This is typically done only via manual annotation, which is subjective and could be heavily influenced by how the vehicle approaches the bump and at what speed. We also base our ground truth on manual annotation, but we ensure that the ground truth is decided by consensus among two or three users. On some sections of the road, we also performed multiple drives to compare the ground truth across drives and arrive at a consensus. The second major challenge with bump detection is that the accelerometer signal is typically of very short duration, on the order of milliseconds. Since a bump results in a significant vertical jerk, we expect it to register in the measurement of a_Z. Also, since the vehicle shifts both up and down, there would be both spikes and dips in a_Z. However, the magnitude of the signal spike can have different characteristics at different speeds.

These can be seen in Figures 6 and 7, which shows a_Z for a car going over a speed bump at low and high speeds, respectively. Note that, at low speeds, there is a distinct dip significantly below $1g$ that is sustained over several samples. We hypothesize that this dip corresponds to the physical phenomenon from when the car's wheel goes over the pothole until the wheel meets the bottom of the pothole. When the wheel impacts the ground, a sharp force is transferred to the vehicle, which is registered as a spike in a_Z. At low speeds, the upward spike is muted, as in Figure 6, while at high speeds, the upward spike can be significant, as in Figure 7. While the sustained dip also occurs during bumps in many of our high speed traces, we also found that it also caused a lot more false positives at high speeds. We suspect that small undulations in the road could be the cause of such false-positive dips.

Motivated by the above observations, we utilize two bump detectors depending on the speed of the

Detector	Accel.	Speed < 25kmph		Speed ≥ 25kmph	
		FN	**FP**	**FN**	**FP**
BUMPY road		40 bumps total		4 bumps total	
z-sus	ACL-1	**25%**	**5%**	50%	0%
	ACL-2	**30%**	**0%**	25%	0%
	ACL-3	**23%**	**5%**	0%	50%
z-peak	ACL-1	28%	15%	0%	125%
(1.45)	ACL-2	20%	5%	0%	125%
	ACL-3	30%	10%	0%	200%
MIXED road		62 bumps total		39 bumps total	
z-sus	ACL-1	**29%**	**8%**	18%	80%
	ACL-3	**37%**	**14%**	0%	136%
z-peak	ACL-1	35%	6%	5%	197%
(1.45)	ACL-3	65%	21%	3%	49%
z-peak	ACL-1	90%	0%	**51%**	**3%**
(1.75)	ACL-3	83%	0%	**41%**	**8%**

Table 5: **False positives/negatives (FP/FN) for bump detectors, *z-peak* derived from [22] and our new *z-sus*. The numbers in bold correspond to the hybrid approach of applying *z-sus* at low speeds and *z-peak* at high speeds.**

vehicle. At high speeds (> 25 kmph), we use the surge in a_Z to detect bumps. This is identical to the *z-peak* heuristic proposed in [22], where a spike along a_Z greater than a threshold is classified as a suspect bump. At low speeds, we propose a new bump detector called *z-sus*, which looks for a *sustained dip* in a_Z, reaching below a threshold T and lasting at least 20 ms (7 samples at our sampling rate of 310 Hz). The choice of the 20 ms duration is motivated by the fact that a car travelling at slow speeds, say 18 kmph or 5 m/s, would travel, in 20 ms, a distance of 10 cm, which roughly corresponds to the radius of a typical pothole. Finally, in order to identify the vehicle speed for applying the appropriate brake detector, we can rely on GSM localization (Section 7) since we only need coarse-grain speed, i.e., whether the vehicle is travelling at low speeds ($<$ 25 kmph) or not.

The two bump detectors, *z-peak* from [22] and our new *z-sus*, were tested over two drives, one along a short section of road approximately 5 km long with a total of 44 bumps or potholes (labeled "bumpy road"), and another approximately 30km long with a total of 101 bumps or potholes from a mixture of bumpy roads interspersed with stretches of smooth highways (labeled "mixed road"). In the case of *z-sus*, a threshold of $T = 0.8g$ was chosen by training over the bumpy road trace and then the same threshold was used over the longer mixed road trace for validation. In the case of *z-peak*, one could use techniques presented in [22] to dynamically tune the threshold to different speeds. However, here we illustrate the best-case scenario for *z-peak* by tuning the threshold parameter separately to its optimal values for low and high speeds, respectively.

During each run, we obtained measurements from two well-oriented accelerometers, ACL-1 and ACL-2, as well as a disoriented accelerometer, ACL-3, that was virtually reoriented. Our goal is to evaluate the bump detectors, both across the different accelerometers and across the two drives. The results are shown in Table 5. Note that, even though we show results for *z-sus* at both low and high speeds for completeness, we wish to reiterate that *z-sus is targeted as a detector only for low speeds.*

We make several observations. First, while we have tuned both the detectors so that the false positive

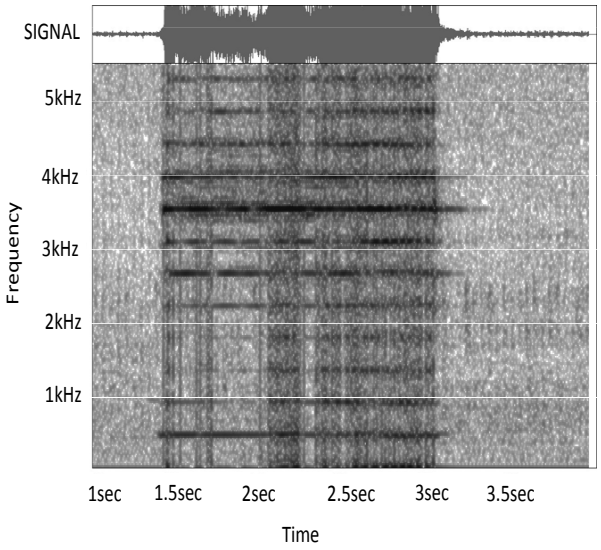

Figure 8: **Spectrogram of the horn of a Toyota Innova minivan**

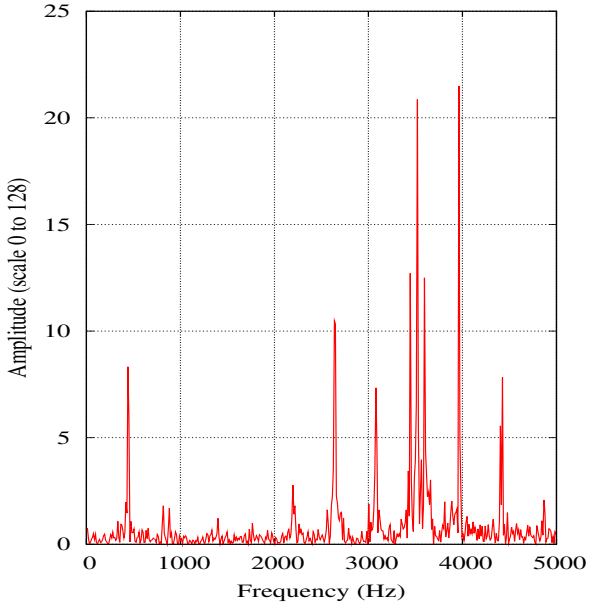

Figure 9: **DFT of horn at 1.6s**

rate is kept low ($< 10\%$), the false negative rate for both the detectors is still quite high (20-30%). This can be explained partly by the difficulty of establishing the ground truth, as noted above. Second, the false positive/negative rates show some variation across all three accelerometers (two well-oriented and one virtually reoriented) for both *z-peak* and *z-sus* but the range of variation is limited for most of the cases. For example, false negative (positive) rates varies between 20-30% (5-15%) for the bumpy road trace. This shows that the virtual reorientation preserves the essential characteristics of the accelerometer signal. Third, consider the case when speed is less than 25 kmph, where *z-peak* performs best with an optimal threshold of $1.45g$. We note that *z-sus* outperforms *z-peak*, with lower false positive rates for attaining similar or better false negative rates. Finally, we note that if *z-peak* is tuned with a threshold of $1.75g$, it performs best at high speeds, achieving 3-8% false positive rates on the mixed road trace (*z-sus* suffers from unacceptably high false positive rates at high speeds).

6 Audio

All phones have a microphone that can serve as a readily-available sensor. In this section, we discuss how the audio sensed through the mobile's microphone can be used for honk detection. Audio sensing does use significant amount of power, though lower than WiFi or GPS (Table 2), and is performed only on an as needed basis (see Section 8). Finally, note that the audio content never leaves the phone; only processed information such as the number of honks detected is sent to the TrafficSense server, alleviating privacy concerns to a large extent.

TrafficSense uses honk detection to identify noisy and chaotic traffic conditions like that at an unregulated intersection. There is considerable work in the field of audio content classification [21, 24, 27] where researchers have used various machine learning approaches to detect, among other things, the sound of a

Phone	Exposed vehicle		Enclosed vehicle	
	FP	**FN**	**FP**	**FN**
KJAM (T=5)	38%	0%	8%	15%
KJAM (T=7)	0%	0%	0%	23%
KJAM (T=10)	0%	19%	0%	54%
iPAQ (T=5)	19%	4%	0%	19%
iPAQ (T=7)	0%	8%	0%	50%
iPAQ (T=10)	0%	27%	0%	81%

Table 6: **False Positives/Negatives (FP/FN) of Honk detector**

horn. Given our need to make detection lightweight so that it can run on a mobile device, we investigate simpler, heuristic-based approaches.

An approach that simply looks for spikes in sound power levels performs quite poorly as it misses a lot of horn sounds that are muted inside an enclosed vehicle and also mistakes any loud noise as a horn. To motivate a more discriminating detector, Figure 8 depicts the spectrogram of a horn sound from 1.5s to 3s, i.e., a frequency versus time plot, with higher sound power depicted by darker shades of grey. The frequency harmonics are clearly visible (with a fundamental frequency under 500Hz) and there is considerable amount of energy around the 3kHz band, the region of highest human ear sensitivity [21].

Thus, we implement a simple detector that performs a discrete Fourier transform on 100ms audio samples and looks for energy spikes. We define a spike as an instantaneous sample that is at least T times the mean, where T ranges between 5 and 10. Based on empirical observations, we arrive at the following heuristic: as long as there are at least two spikes, including at least one spike in the 2.5 kHz to 4 kHz region corresponding to the region of highest human ear sensitivity, we classify the audio sample as corresponding to a honk. Figure 9 plots the sound amplitude versus frequency based on a discrete Fourier transform performed on a window of 1024 audio samples (with an audio sampling rate of 11025 Hz) at time 1.6s of the horn sample depicted in Figure 8. The spikes match well with the darker shades in the spectrogram and would indicate a honk per the heuristic noted above.

To evaluate the performance of our honk detector, we use audio traces collected using four phones simultaneously at a chaotic and noisy traffic intersection. We use two HP iPAQs and two i-Mate KJAMs. One phone of each type was placed inside an enclosed vehicle and one of each type placed outside the vehicle (representing an exposed vehicle such as a motorbike). We establish the ground truth by listening to the audio clip from the phone placed outside the vehicle and manually identifying the honk times, with a granularity of 1 second. We then use the honk detector with different spike thresholds, T, to automatically identify honks and compare it with the ground truth. The audio trace was about 100 seconds in duration and had 26 honks from a variety of vehicles.[2]

The results are shown in Table 6. Assuming a null hypothesis of no horn, the detector suffers false negatives, i.e. missed honk detection, mostly when the spikes in the 2.5-4kHz band are insignificant, while it suffers false positives mostly when the spike threshold is low enough for other sounds to be mis-identified as honks.

We make three observations based on these results. First, with a spike threshold that is large enough to avoid false positives, the honk detector performs better (i.e., has fewer false negatives) in the exposed vehicle scenario due to the higher sound power level. Second, the varying sensitivity of microphone in

[2]The need to manually establish the ground truth by listening limits the length of the trace that is feasible to evaluate.

different phones can result in different number of honks being detected even on the same trace. These two observations indicate that when comparing honk data at different locations, care must be taken to separate out the data based on phone-type and ambient noise level before comparison. Third, a high spike threshold can substantially guard against false positives (0% for $T \geq 7$ in this trace), though, it cannot eliminate it completely. In one of our audio traces, there was an instance of a spurious detection of a horn arising from the sound of a bird chirping that displayed horn-like characteristics. In general, this detector will have false positives whenever the audio content has honk-like spectrogram characteristics (e.g. alarms). If these false-positives become significant in some settings, a more sophisticated honk detector would be necessary.

Finally, the honk detector is very efficient. Our implementation running on the iPAQ phone consumes only about 58ms of CPU time to detect honks in 1 second worth of 16bit, 11025 Hz audio samples (i.e., CPU utilization of 5.8%). Note that the CPU utilization would be further reduced in practice because honk detection would be triggered on-demand rather than be run continuously, as discussed in Section 8.

7 Localization

Localization is a key component of TrafficSense, as it is in any sensing application. Each phone participating in TrafficSense would need to continuously localize its current position, so that sensed information such as honking or braking can be tagged with the relevant location. Thus, an energy-efficient localization service is a key requirement in TrafficSense.

As discussed in Section 2, there are a variety of approaches for outdoor localization including using Global Positioning System (GPS), WiFi [19], and GSM [18, 31]. However, given the high energy consumption characteristics of the WiFi and GPS radios (see Table 2), these are not suitable for continuous localization in TrafficSense. Thus, we primarily rely on using GSM radios for energy-efficient coarse-grain localization and, as discussed in Section 8, we trigger the use of fine-grain localization using GPS when necessary.

There has been some recent work on using GSM to perform localization in indoor [31] and outdoor environments [18]. In [18], authors show that GSM signal strength-based localization algorithms can be quite accurate, with median errors of $94m$ and $196m$ in downtown and residential areas, respectively, around Seattle. When we tried to apply these techniques to the GSM tower signal data we collected in Bangalore, we found that the characteristics of the data was significantly different (much more heterogeneity) that these prior techniques were not applicable.

We use four HTC Typhoon phones, subscribed to two different GSM service providers (two phones per provider), to collect GSM signal information. Each phone records the following six-tuple (MCC, MNC, $LACID$, $CELLID$, $ARFCN$, $RSSI$) for the strongest seven GSM signals that it sees every second. The MCC, MNC are the mobile country and network codes that identify a GSM operator, $LACID, CELLID$ are the location area and cell IDs that identify a cell tower, $ARFCN$ identifies the GSM carrier frequency of the signal, and $RSSI$ is the received signal strength value of the signal. Since the first five fields of the six tuple are unique to a particular GSM signal, we call this five tuple[3] *tower ID*. Let a set of $1 \leq n \leq 7$ tower IDs used for identifying a location be called a *tower signature*.

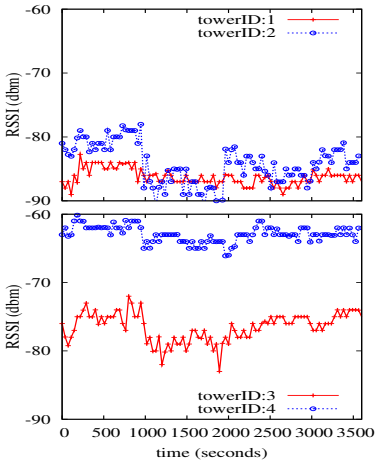

Figure 10: **Seattle, Phone 1**

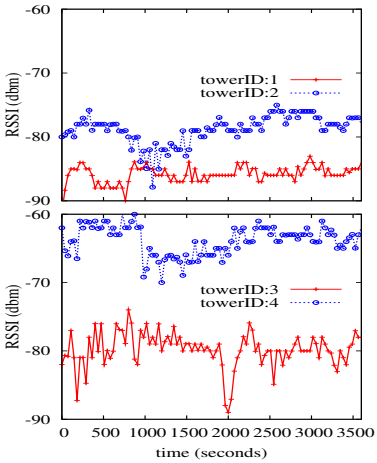

Figure 11: **Seattle, Phone 2**

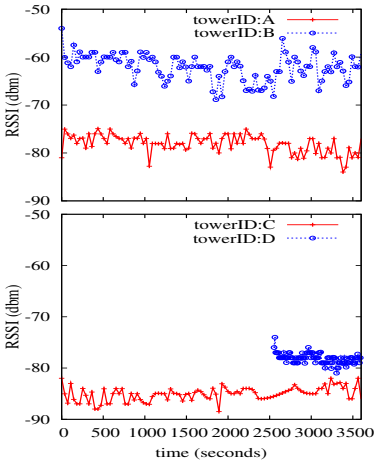

Figure 12: **Bangalore, Phone 1**

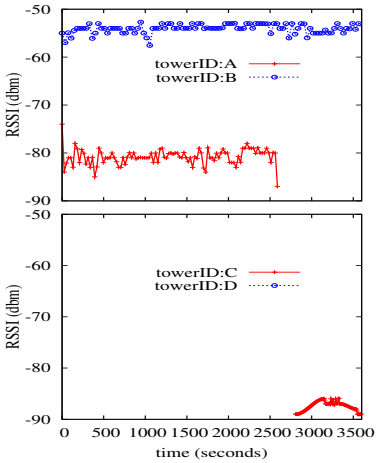

Figure 13: **Bangalore, Phone 2**

7.1 GSM signal characteristics in Seattle and Bangalore

In Figure 10, we present the *RSSI* variation, in a one hour trace, of four most dominant (in terms of number of occurences) tower IDs of a *static* GSM phone in a Seattle neighborhood. Figure 11 presents the same for another *static* GSM phone placed physically adjacent to the previous phone and subscribed to the same wireless operator. From these graphs, we can see that (1) there is up to 10dbm variability of RSSI for a single tower ID over time, and (2) there is up to 20dbm variability of RSSI between the same tower ID of two phones subscribed to the same operator. The RSSI variability between the different phones is one potential problem for the RSSI-based localization algorithms proposed in [18]. Specifically, maintaining a tower signature database for each {mobile phone model, operator} pair would be expensive.

[3]Technically, the first four fields are sufficient to identify a particular GSM cell sector. However, we found that including the ARFCN improved the accuracy of all our algorithms, implying possibly that different frequencies are transmitted at different power levels.

A second problem for RSSI-based localization approaches is apparent when we examine a similar trace, collected in Bangalore. Figure 12 presents RSSI vs time for two *static* phones in Bangalore. While the RSSI variability over time is similar to the Seattle trace, we find that one of the top four dominant tower IDs in phone 1 does not even appear for the entire hour in phone 2. Looking at the trace of the second phone in Figure 13, we find that one of the tower IDs ('D') is not even seen by this phone and two tower IDs ('B','C') are seen for much shorter durations. The primary reason for the difference between the Seattle and Bangalore traces is the number of unique tower IDs seen in the respective traces over the one hour duration. In the Seattle traces, we found 28 and 17 tower IDs, while, in the Bangalore traces, we found 93 and 62 tower IDs. We believe that the significantly higher number of unique tower signatures seen in the Bangalore trace is due to the higher density of GSM users in India. For example, the inter-tower spacing in India is estimated to be 100 meters compared to the typical international micro-cell inter-tower spacing of about 400 meters [33].

The high density of towers seen in the Bangalore trace creates significant difficulties for prior RSSI-based localization algorithms. For example, the RADAR-based fingerprint algorithm, which outperformed other algorithms considered in [18] when applied to the traces from Seattle, requires that the tower signature (a set of, upto 7, tower IDs) seen by the phone at a given time exactly match the tower signature stored in the database. The matched database entries that are the closest in signal strength space are then used for localization. In the above Bangalore trace, given 93 available towers and a phone exposing only the strongest 7 tower IDs at any given time, the probability of an exact match of a current signature with that in the database is quite small (a 4% probability of match, based on training data from 23 drives over 12 days in Bangalore), making these algorithms ineffective.

7.2 Localization Algorithms

We now present two localization algorithms that cater to our observations of GSM signals for a setting like Bangalore, namely, the high density and the varying set of visible tower IDs. The high density of towers suggests that even simple localization algorithms, such as the strongest signal approach used for example in indoor, dense WiFi settings [16], may be effective. Thus, our first localization algorithm, *strongest signal (SS)*, relies on a training database that maps the tower ID with the strongest-signal to an average latitude/longitude position. During localization, the phone simply looks up its current strongest tower ID[4] in the training database and returns the corresponding latitude/longitude position as its estimate. We also evaluate a generalization of the SS algorithm, which includes all towers whose signal is stronger than a threshold as part of the tower signature. Based on our traces, we find that the probability of finding a match with SS is 96% compared to 4% for the exact match approach, as indicated above.

If the training database is sparse or out-of-date (i.e., it may not include most or all of the strongest signal tower IDs at a given location), then the strongest signal tower-based localization will not be robust, i.e., the given strongest signal tower ID may not have a match in the database. In order to increase robustness, we would like to be able to match the (largest) subset of towers seen currently to the one in the database. Such an approach will fail to localize only if the training database does not have even one of the seven tower IDs in a given reading. This leads to our second localization algorithm, the *Convex Hull Intersection (CHI)*. During training, each visible tower ID is mapped to a convex hull of the latitude/longitude positions from which the tower ID can be seen. During localization, the phone looks up each of the seven tower IDs in the database, obtains their respective convex hulls if available, performs an intersection of the convex hulls[5], and

[4]Note that the strongest tower ID is not necessarily the current attached tower, that most phones expose.

[5]There are 2^n ways of intersecting n convex hulls, with potentially different results due to incomplete training data. We execute a subset, n, of 2^n possibilities, and choose the one that results in an intersection with the smallest radius. In cases

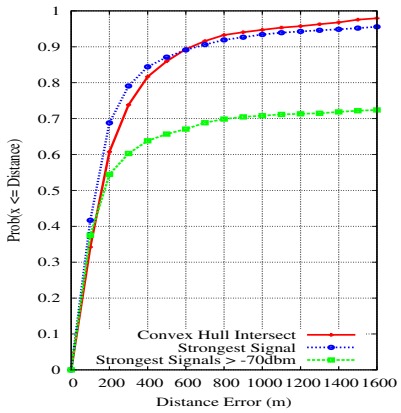

Figure 14: **Bangalore trace**

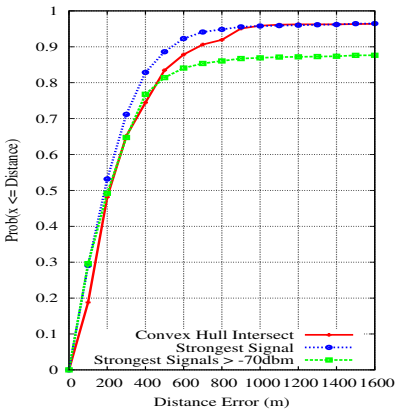

Figure 15: **Seattle trace**

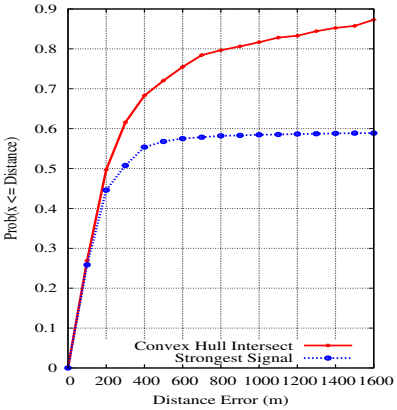

Figure 16: **Bangalore, 2 months later**

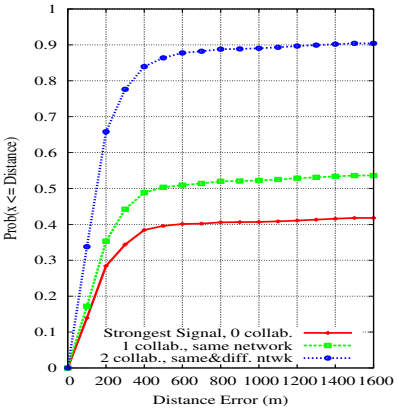

Figure 17: **Collaboration**

returns the centroid of the resulting intersection as its location estimate.

7.3 Evaluation

We compute the localization distance estimate error of the SS and CHI algorithms using a rich collection of cellular traces from Bangalore. We use 23 drives over 12 days in Bangalore as training data for building the tower ID database and use 10 drives over 5 days for validation (see Table 1). The training data covers a significant portion of roads in the heart of Bangalore city (see Figure 1) while the validation drives comprises a subset of these roads. We use GPS-based localization data, collected during the same drives, as the ground truth for computing the error in our location estimates.

Figure 14 plots the cummulative distribution function (CDF) of the localization distance estimate error of the SS and CHI algorithms using a rich collection of cellular traces from Bangalore (see Table 1). We

where the intersection of the full set of tower IDs result in a null intersection, we choose the smallest non-null intersection set.

use GPS-based localization as the ground truth for computing our estimate errors. In addition to the SS algorithm that only picks the single strongest tower as part of the signature, we also plot the performance of generalized SS algorithm, by only including the set of all tower IDs with $RSSI > -70dbm$ as part of the tower signature. The CHI algorithm selects all visible tower IDs with $RSSI > -80dbm$ as its tower signature, thereby filtering out spurious tower sightings with very weak RSSI. From the figure, we can see that (1) the simple SS algorithm with one tower performs well with median localization error of only $117m$; (2) Using more than one tower with SS significantly reduces the probability of localization to 73%; (3) CHI is a little worse than SS with a median error of $155m$ but performs better at 90th percentile ($626m$ vs $660m$) and higher. While our focus is on localization in areas covered by dense GSM networks, for comparison Figure 15 plots the CDF of localization errors using our limited Seattle area trace. While the overall shape of the curves is similar to Figure 14, we observe that (1) the median errors are generally higher ($179m$ vs $117m$ for SS), and (2) using more towers does not reduce the localization probability as significantly as in the Bangalore trace (88% vs 73%), thereby allowing the use of more sophisticated signal strength-based algorithms.

In order to test the robustness of these algorithms, we gathered traces from a few drives again in November 2007 and tried to perform localization on these drives using training data based on traces collected in September. Figure 16 plots the CDF of the localization distance errors. In this case, the CHI algorithm significantly outperforms the SS algorithm as it is much more resilient to the mix of tower IDs visible at a given time.

After analyzing the trace, we found that one of our two service providers had made significant changes to their network in the preceding two months. In such a setting, collaboration with neighboring devices can be quite effective. Figure 17 presents the CDF for the SS algorithm augmented with collaboration with 0, 1,and 2 neighboring phones, that simply transmit their visible cell tower information periodically using Bluetooth or WiFi. When a neighboring phone using a different operator is available, the performance of the SS algorithm is dramatically improved with probability of localization increasing from almost 40% to 90%.

7.4 Speed Estimation

Traffic speed monitoring is a direct application of localization. Using the SS algorithm, the median and 90^{th} percentile error in the speed estimate over 100 second travel segments in the Bangalore trace were 3.4 kmph and 11.2 kmph, respectively. The median and 90^{th} percentile relative error in the speed estimates were 21% and 70% respectively, thus providing a reasonable estimate of traffic speed in the fairly slow moving peak-hour traffic in Bangalore. The accuracy in absolute terms (the 3.4 kmph and 11.2 kmph figures noted above) is arguably more important for it would determine our ability to distinguish between traffic that is crawling on a congested road and moving freely on a fast road. In comparison, the (median, 90^{th} percentile) absolute and relative speed estimate errors were (6.6 kmph, 18.5 kmph) and (17%, 87%) in our Seattle trace.

7.5 Summary

In summary, given the density of cell tower locations in the cities of the developing world, a simple strongest signal-based algorithm can provide accurate energy-efficient localization and speed estimates, provided the training data is rich and recent. On the other hand, the convex hull intersection-based algorithm provides better robustness if the training data is sparse/stale. These algorithms result in a median localization distance error of $120 - 150m$ and median absolute and relative speed estimate errors of $3.4kmph$ and 21%,

Item	Median error	90^{th} percentile error
Localization Distance	117m	660m
Absolute speed	3.4 kmph	11.2 kmph
Relative speed	21%	70%

Table 7: **Localization and Speed estimate errors for SS**

respectively in our Bangalore traces. The results for the Bangalore validation traces are summarized in Table 7.

8 Triggered Sensing

Besides making efficient use of individual sensors, there is the opportunity for energy savings by employing sensors in tandem. A low-energy but possibly less accurate sensor could be used to trigger the operation of a high-energy and possibly more accurate or complementary sensor only when needed. Of the sensors in our current prototype, we deem the GSM radio and the accelerometer as low-energy sensors, and GPS and the microphone as high-energy sensors (see Table 2).

The GSM radio and the accelerometer are always kept on. These are then used to trigger GPS and/or the microphone when needed. Specific instances of triggered sensing in TrafficSense include the following:

Localization: GSM-based localization is used to trigger GPS to obtain an accurate location fix on a feature of interest. Consider a phone that detects a major pothole in the road using its accelerometer measurements (Section 5.4.3). Since it uses GSM-based localization by default, the phone has an approximate idea of the location of this pothole. The next time the phone (or another participating phone, assuming information is shared across phones) finds itself in the same vicinity based on GSM information, it triggers GPS so that an accurate location fix is obtained on the pothole. The energy savings can be very significant, depending on the specific setting. For example, if the GPS is triggered when the GSM location estimate is within 500 m of the desired latitude/longitude and turned off when the desired location has been passed, on a 20 km long drive, GPS would need to be turned on only 3.2% of the time (averaged over 10 runs).

Virtual reorientation: The accelerometer (Section 5.2.2) and also user activity detection (Section 5.3) is used to determine that a phone's orientation may have changed and hence the virtual reorientation procedure needs to be repeated. GPS is then triggered at such a time to help detect the braking episodes needed for reorientation (Section 5.2.3). We could optimize this further by using the GSM/accelerometer to determine that the phone is likely in a moving vehicle (based on changes in the GSM/accelerometer measurements) before triggering GPS.

Honk detection: Braking detection (Section 5.4.1) could be used to trigger honk detection. If significant levels of braking as well as honking are detected, it might point to traffic chaos.

We defer an evaluation of these triggered sensing techniques to future work.

9 Implementation Status

We have implemented all of the algorithms described in the paper, most of these on the HP iPAQ smartphone running Windows Mobile 5. The virtual reorientation algorithm runs on the HP iPAQ by gathering measurements from the Sparkfun WiTilt accelerometer via a serial port interface over its Bluetooth radio.

The algorithm is implemented in Python and C#. The audio honk detection algorithm also runs on the HP iPAQ. This is implemented in C# and invokes an FFT library for the discrete fourier transform and the Windows Mobile 5 `coredll` library for capturing the microphone input.

The GSM-localization algorithms are the only ones that do not currently run on the iPAQ, since the iPAQ does not expose the necessary cell tower information. The cell tower information used in our traces is obtained on the rebranded HTC Typhoon phones based on reading a fixed memory location, that has been obtained via reverse-engineering and is well-known [18]. The localization algorithms are currently implemented in perl and can easily be ported to the iPAQ, if and when the necessary cell tower information becomes accessible. At this time, we have implemented a simple C# program to obtain GPS information from the GPS receiver on the iPAQ and use this for our localization needs.

Finally, we have also implemented the detectors for identifying user interactions, thereby invalidating accelerometer and microphone-based sensing. This implementation is on Windows Mobile 5, which provides the necessary hooks to intercept keypad and mouse events, and also access to call log information for both incoming and outgoing calls.

10 Conclusion

TrafficSense is a system for rich monitoring of road and traffic conditions using mobile smartphones equipped with an array of sensors (GPS, accelerometer, microphone) and communication radios. In this paper, we have focused on the sensing component of TrafficSense, specifically on how these sensors and radios are used to detect bumps and potholes, braking, and honking, and to localize the phone in an energy-efficient manner. We have presented techniques to virtually reorient a disoriented accelerometer and to use multiple sensors in tandem, with one triggering the other, to save energy. Our evaluation on extensive drive data gathered in Bangalore has yielded promising results. We have a prototype of TrafficSense, minus GSM-based localization, running on Windows Mobile 5.0 Pocket PCs.

Acknowledgements

Several people wrote useful pieces of code that we have borrowed in our prototype of TrafficSense: Ramya Bharathi Nimmagadda (detecting user interaction with a phone), Nilesh Mishra (extracting GSM tower information), and Pei Zhang (a rudimentary TrafficSense server). The drivers of Japan Travels as well as Kannan and Srinivas endured several hours of driving experiments with patience. Ranjita Bhagwan and Geoff Voelker provided insightful comments on an earlier draft that have helped improve this paper. We thank them all.

References

[1] AirSage Wireless Signal Extraction (WiSE) Technology. http://www.airsage.com/wise.htm.

[2] Bangalore Transport Information System. http://www.btis.in/.

[3] California Center for Innovative Transportation: Traffic Surveillance. http://www.calccit.org/itsdecision/serv_and_tech/Traffic_ Surveillance/surveillance_overview.html.

[4] Freescale MMA7260Q Accelerometer. http://www.sparkfun.com/datasheets/Accelerometers/MMA7260Q-Rev1.pdf.

[5] HP iPAQ hw6965 Mobile Messenger Specifications. http://h10010.www1.hp.com/wwpc/au/en/ho/WF06a/1090709-1113753-1113753-1113753-1117925-12573438.html.

[6] INRIX Dynamic Predictive Traffic. http://www.inrix.com/technology.asp.

[7] Intelligent Transportation Systems. http://www.its.dot.gov/.

[8] Mobile boom helps India reach Internet goal before time. http://economictimes.indiatimes.com/articleshow/2341785.cms

[9] OnStar by GM. http://www.onstar.com/.

[10] SmartTrek: Busview. http://www.its.washington.edu/projects/busview_ overview.html.

[11] SparkFun Wireless Accelerometer/Tilt Controller version 2.5. http://www.sparkfun.com/commerce/product_info.php?pr 254.

[12] Telecom Regulatory Authority of India (TRAI) Press Release, Oct 2007. http://www.trai.gov.in/trai/upload/PressReleases/ 506/pr22oct07no91.pdf.

[13] Worldwide Mobile Subscriptions, Informa Telecoms & Media, July 2007. http://www.informatm.com/itmgcontent/icoms/whats-new/20017437800.html.

[14] D. Burke et al. Participatory Sensing. In *World Sensor Web Workshop*, 2006.

[15] L. Buttyan and J.-P. Hubaux. *Security and Cooperation in Wireless Networks*. Cambridge University Press, 2007.

[16] R. Chandra, J. Padhye, A. Wolman, and B. Zill. A Location-based Management System for Enterprise Wireless LANs. *NSDI*, 2007.

[17] R. Chang, T. Gandhi, and M. M. Trivedi. Vision Modules for a Multi-Sensory Bridge Monitoring Approach. In *IEEE Intelligent Transportation Systems Conference*, 2004.

[18] M. Chen, T. Sohn, D. Chmelev, D. Haehnel, J. Hightower, J. Hughes, A. LaMarca, F. Potter, I. Smith, and A. Varshavsky. Practical Metropolitan-Scale Positioning for GSM Phones. In *Ubicomp*, 2006.

[19] Y. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy Characterization for Metropolitan-scale Wi-Fi Localization. In *MobiSys*, 2005.

[20] D. J. Dailey. SmartTrek: A Model Deployment Initiative. Technical report, May 2001. Report No. WA-RD 505.1, Washington State Transportation Center (TRAC), University of Washington, http://www.wsdot.wa.gov/Research/Reports/500/505.1.htm.

[21] D. Ellis. Detecting alarm sounds. In *CRAC workshop, Aalborg, Denmark*, 2001.

[22] 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 *MobiSys*, 2008.

[23] H. Goldstein. *Classical Mechanics, 2nd Edition*. Addison-Wesley, 1980. (Section 4.4 on 'The Euler Angles', pp. 143–148).

[24] D. Hoiem, Y. Ke, and R. Sukthankar. SOLAR: Sound Object Localization and Retreival in Complex Audio Environments. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2005.

[25] B. Hull, V. Bychkovsky, K. Chen, M. Goraczko, A. Miu, E. Shih, Y. Zhang, H. Balakrishnan, and S. Madden. The CarTel Mobile Sensor Computing System. In *SenSys*, 2006.

[26] W. D. Jones. Forecasting Traffic Flow. *IEEE Spectrum*, Jan 2001.

[27] H. Kim, N. Moreau, and T. Sikora. Audio Classification Based on MPEG-7 Spectral Basis Representations. *IEEE Transactions on Circuits and Systems for Video Technology*, May 2004.

[28] A. Krause, E. Horvitz, A. Kansal, and F. Zhao. Toward Community Sensing. In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2008.

[29] J. Krumm and E. Horvitz. Predestination: Where Do You Want to Go Today? *IEEE Computer Magazine*, Apr 2007.

[30] C.-T. Lu, A. P. Boedihardjo, and J. Zheng. AITVS: Advanced Interactive Traffic Visualization System . In *ICDE*, 2006.

[31] V. Otsason, A. Varshavsky, A. LaMarca, and E. de Lara. Accurate GSM Indoor Localization. In *Ubicomp*, 2005.

[32] A. Rahmati and L. Zhong. Context for Wireless: Context-sensitive Energy-efficient Wireless Data Transfer. In *MobiSys*, 2007.

[33] T. V. Ramachandran. How should spectrum be allocated? *The Economic Times*, Nov 2007. http://economictimes.indiatimes.com/Debate/How_should_spectrum_be_allocated/articleshow/2520807.cms.

[34] B. L. Smith, H. Zhang, M. Fontaine, and M. Green. Cellphone Probes as an ATMS Tool. Technical report, Jun 2003. Smart Travel Lab Report No. STL-2003-01, University of Virginia, http://ntl.bts.gov/lib/23000/23400/23431/CellPhoneProbes-final.pdf.

[35] A. Varshavsky. Are GSM Phones THE Solution for Localization? In *WMCSA*, 2006.

[36] E. W. Weisstein. Euler Angles. *MathWorld – A Wolfram Web Resource*. http://mathworld.wolfram.com/EulerAngles.html.

[37] J. Yoon, B. Noble, and M. Liu. Surface Street Traffic Estimation. In *MobiSys*, 2007.

Appendix A: Estimating the Angle of Pre-rotation

Since the only acceleration experienced when stationary or in steady motion is along Z, $a_X = 0$ and $a_Y = 0$. For a well-oriented accelerometer, we would also have $a_x = 0$ and $a_y = 0$. However, for a disoriented accelerometer, the pre-rotation followed by the tilt implies that x and y would, in general, no longer be orthogonal to Z, so a_x and a_y would be equal to the projections of the $1g$ acceleration along Z onto x and y, respectively. To calculate this, we first consider the pre-rotation and decompose each of a_x and a_y into their components along X and Y, respectively. Then when the tilt (about Y) is applied, only the components of a_x and a_y along X would be affected by gravity.

Thus after the pre-rotation and the tilt, we have: $a_x = cos(\phi_{pre})sin(\theta_{tilt})$ and $a_y = sin(\phi_{pre})sin(\theta_{tilt})$. So $tan(\phi_{pre}) = \frac{a_y}{a_x}$, which yields Equation 2 for estimating ϕ_{pre}.

Appendix B: Estimating the Angle of Post-rotation

We can compute $a_X^{'}$ by running through the steps of pre-rotation, tilt, and post-rotation in sequence, at each step applying the decomposition method used in Appendix A. Starting with just pre-rotation, we have $a_X^{'pre} = a_x cos(\phi_{pre}) + a_y sin(\phi_{pre})$ and $a_Y^{'pre} = -a_x sin(\phi_{pre}) + a_y cos(\phi_{pre})$. After tilt is also applied, we have $a_X^{'pre-tilt} = a^{'pre} cos(\theta_{tilt}) - a_z sin(\theta_{tilt})$ and $a_Y^{'pre-tilt} = a_Y^{'pre}$ ($a_Y^{'pre-tilt}$ remains unchanged because the tilt is itself about Y). Finally, after post-rotation is also applied, we have $a_X^{'} = a_X^{'pre-tilt-post} = a_X^{'pre-tilt} cos(\psi_{post}) + a_Y^{'pre-tilt} sin(\psi_{post})$. Expanding this, we have $a_X^{'} = [(a_x cos(\phi_{pre}) + a_y sin(\phi_{pre}))cos(\theta_{tilt}) - a_z sin(\theta_{tilt})]cos(\psi_{post}) + [-a_x sin(\phi_{pre}) + a_y cos(\phi_{pre})]sin(\psi_{post})$.

To maximize $a_X^{'}$ consistent with a period of sharp deceleration, we set its derivative with respect to ψ_{post}, $\frac{da_X^{'}}{d\psi_{post}}$, to zero. So $-[(a_x cos(\phi_{pre}) + a_y sin(\phi_{pre}))cos(\theta_{tilt}) - a_z sin(\theta_{tilt})]sin(\psi_{post}) + [-a_x sin(\phi_{pre}) + a_y cos(\phi_{pre})]cos(\psi_{post}) = 0$, which yields Equation 3 for estimating ψ_{post}.