FullStop: Tracking Unsafe Stopping Behaviour of Buses

Ravi Bhandari
Indian Institute of Technology Bombay

Bhaskaran Raman
Indian Institute of Technology Bombay

Venkata N. Padmanabhan
Microsoft Research India

***Abstract*—Road safety is a critical issue the world-over, and the problem is particularly acute in developing countries, where the combination of crowding, inadequate roads, and driver indiscipline serves up a deadly cocktail. We believe that mobile devices can play a positive role in this context by detecting dangerous conditions and providing feedback to enable timely redressal of potential dangers. This paper focuses on a specific problem that is responsible for many accidents in developing countries: the stopping behaviour of buses especially in the vicinity of bus stops. For instance, buses could arrive at a bus stop but continue rolling forward instead of coming to a complete halt, or could stop some distance away from the bus stop, possibly even in the middle of a busy road. Each of these behaviours can result in injury or worse to people waiting at a bus stop as well as to passengers boarding or alighting from buses.**

We present FullStop, a smartphone-based system to detect safety risks arising from bus stopping behaviour, as described above. We show that the GPS and inertial sensors are unable to perform the fine-grained detection needed, by themselves. Therefore, FullStop is based on the view obtained from looking out to the front of the vehicle using the camera of a smartphone that is mounted on the front windshield. Using optical flow vectors, with several refinements, FullStop running on a smartphone is able to effectively detect safety-related situations such as a rolling stop or stopping at a location that is displaced laterally relative to the designated bus stop.

I. Introduction

Road safety is a critical issue worldwide. Road accidents caused an estimated 1.25 million fatalities worldwide in 2013, placing it in the top 10 causes of death worldwide [1]. This is particularly severe in developing countries, with a rapidly growing population of vehicles, inadequate road infrastructure, and poor driver training and discipline. For instance, India alone had almost 240,000 road fatalities in 2013, at a rate of 130 fatalities per 100,000 vehicles, while China had a similarly high 105 fatalities per 100,000 vehicles, far higher than the U.K. at 5, the U.S. at 13, and Singapore at 20 [1].

In this paper, we focus on a particular problem: the stopping behaviour of buses. Buses are notorious in many cities for the accidents they cause, with them even earning the unenviable sobriquet "killer buses" in cities such as New Delhi [2]. A big part of the problem is that the drivers are often in a hurry, whether due to impatience or because of drive distance quotas they are expected to fulfil, even as increasing congestion makes it difficult for them to do so [2]. So drivers often cut corners.

This is more evident at bus stops, where drivers often engage in a variety of unsafe behaviours, endangering the lives of passengers [3]. Drivers could: **S1:** Perform a rolling stop, wherein the bus keeps rolling forward even as passengers are alighting or boarding. **S2:** Stop away from the bus stop laterally, i.e., in the middle of the road rather than in the leftmost lane (we assume without loss of generality that vehicles drive on the left-side of the road). **S3:** Stop away from the bus stop longitudinally, either stopping short or overshooting it, causing people waiting for the bus to rush towards it.

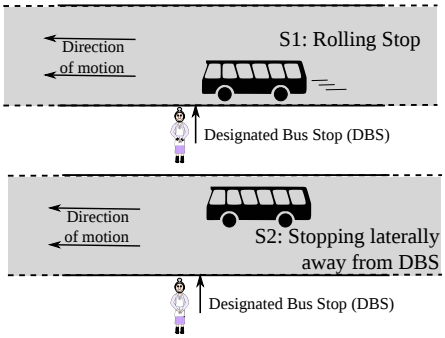

Fig. 1. Unsafe stopping behaviours

The above issues are depicted in Fig. 1. Note that a "bus stop" is often just a designated spot by the side of the road itself rather than having a separate turn-out off the road for buses to pull in to.

With the proliferation of sensor-rich smartphones, there is a growing body of work on using these devices to monitor driving, to detect such behaviours as speeding, sharp braking, cornering, etc.. Our work here derives inspiration from this body of work but focuses specifically on unsafe stopping behaviour, which to our knowledge has not been studied before. To this end, we present FullStop, a smartphone-based system that uses the camera to monitor the stopping behaviour of buses. This paper focuses on situations S1 and S2, while S3 is present in our technical report [4].

The key challenge in detecting S1 and S2 lies in the fine granularity of sensing required. For instance, consider the case of a bus stopping at left extreme of the road, as it should, versus it stopping one lane over to the right. These two cases are poles apart from a safety viewpoint but may only represent a difference of about 3m in location, too small to be reliably detected using GPS. Likewise, a bus that is rolling forward at say 5 Kmph would pose a significant safety risk versus one that has fully stopped. However, as we show later, detecting rolling at such low speed is challenging with GPS or an accelerometer sensor. Further, GPS accuracy worsens in urban canyons.

To address these challenges, FullStop uses the smartphone's

978-1-5386-1182-1/18/$31.00 ©2018 IEEE

camera, which is positioned to look out from the front of the vehicle. At its core, our camera-based detection employs optical flow computation to detect (a lack of) movement or a shift in position even to a slight degree (e.g., a slow roll or a slight shift in position). However, we have had to do much fine-tuning to fit the specific problem context, as we elaborate in later sections. We also make the following assumptions: **A1:** The GPS coordinates of the designated bus stops (DBS) along the bus route are known. **A2:** Instead of being recessed, the bus stops could be at the left edge of a road itself, as is the case in many cities.

We mention the related work in Section II and then discuss the design of FullStop in detail in Section III. We then evaluate FullStop, both via controlled experiments and experiments in the wild in Sec. IV. Our results, based on over 140km of smart-phone video data collected on city buses, over several days and a variety of road scenarios, show FullStop can detect the stop-of-bus event (S1) with low false positive rate (FPR) and false negative rate (FNR) of 10-15%, a significant improvement over detection using the GPS or accelerometer. The lateral displacement (S2) detection algorithm too performs with FPR and FNR of under 10%, even in challenging city road conditions. Based on the data from the city buses, we also quantify the occurence of unsafe stopping behaviours noted above and show that these are indeed quite common.

II. Related Work

Smartphones and similar hardware has been used in prior work in various road transportation related applications. Nericell [5] uses smartphones to detect various road and traffic events such as potholes, honking, etc. Pothole patrol [6] too uses sensors such as the accelerometer to detect road anomalies, though using a dedicated measurement box. The work in [7] uses smartphone-based crowd-sourcing for prediction of bus arrival time. While our work is related to these in terms of the use of a smartphone or similar sensing hardware, we focus on road safety aspects, specifically the stopping behaviour of buses, not considered thus far.

Camera-based sensing has also been used for traffic-related applications. Signal Guru [8] used a dashboard-mounted smartphone to detect traffic signals and deduce their timing, to make traffic flow smooth. The iOnRoad app [9] uses a similarly-mounted camera to do ranging with respect to the vehicle in front, assuming well-marked, fixed-width lanes. Other work has used external cameras to measure traffic; e.g., [10] does so in a developing regions context, with heterogeneous traffic and chaotic road conditions. While our work also leverages the camera of a windshield-mounted smartphone, our specific application is quite different. In particular, we use camera-based sensing to detect movement and displacement that is too fine to be reliably detected with other sensors.

There has also been much work focusng on using sensors for improving road safety. High-end cars and self-driving cars [11] include an array of sensors (e.g., RADAR, LIDAR) and communication technologies such as DSRC and VANETs [12] to improve safety. Of greater interest to us is work on smartphone-based systems for improving safety. [13] uses a smartphone's inertial sensors to detect unintentional lane departures of a moving vehicle. CarSafe [14] uses the front and back camera, respectively, to monitor external conditions (e.g., following distance, lane trajectory) and also internal (e.g., the driver's state). This body of work inspires our work, although both our application (stopping behaviour of buses) and the setting (developing regions) are different. For instance, with regard to lane position of a stopping event, our approach (Section III-D) can depend on neither lane markers being present nor there being measurable movement (since the bus would have stopped).

Finally, we also note that there is growing interest in insurance telematics using smartphones and/or other dedicated sensors [15], [16]. There are a number of start-ups focused on measuring driving for this purpose [17]–[20]. However, to the best of our knowledge, none of this work considers the problem of the stopping behaviour of buses, perhaps because economics means that these start-ups are focusing on the developed regions where this problem does not quite exist.

III. Design of FullStop

We now describe the development of algorithms in FullStop, aimed at detecting situations S1 and S2. Our algorithm development has undergone many iterations, guided in each step by experimental observations. Hence, we begin with a description of our experimental data (Sec. III-A), and thereafter describe the various algorithms (Sec. III-B-Sec. III-D).

A. Data Collection

To develop and evaluate our various algorithms, we have collected extensive video data on buses plying on various bus routes in Mumbai. To mimic a real deployment setting, a smartphone is mounted on the front-left windshield of a bus (the driver sits on the front-right), with its back camera looking out to the front of the bus. Since the phone cannot be left unattended because of concerns regarding its physical safety, we use a strong adhesive tape that helps hold the phone firmly in place during data collection while also permitting us to remove it easily at the end of the bus trip.

Since the set of visible features will likely vary significantly between day and night, we collect data during both day and night periods. Our algorithms assume that the buses ply on city roads which are well lit at night time. This is true in most scenarios (see Fig. 5), especially near designated bus-stops where people wait.

As summarized in Table I, the data collected represents a wide variety of roads, bus routes, traffic conditions, and time-of-day. During these runs, our mobile app collected not only video data but also GPS and accelerometer sensor readings.

Ground Truth Marking: To evaluate our algorithms, we need two kinds of ground truth. First, in the case of a stop-of-bus event (SBE), we need to know the time instant when the bus stopped and when it started moving again, i.e., the start and end of each SBE. Second, we need to know the locations of the designated bus stops (DBS) on each route,

TABLE I
SUMMARY OF DATA COLLECTED ON BUSES

	Day	Night
Number of trips	22	10
Avg trip length (km)	5.1	3.5
Total distance (km)	106.3	34.7
Avg trip duration (hr:min)	0:24	0:23
Total duration (hr:min)	8:29	3:48
Avg DBSs/trip	13.6	9.0
Total DBSs	285	90
Total unique DBSs	26	20
Avg SBEs/trip	21.1	18.7
Total SBEs	443	187

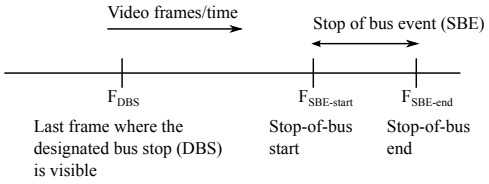

Fig. 2. Designated bus stop (DBS), stop-of-bus event (SBE)

and for each bus stop, where with respect to the DBS the bus actually stopped (if it did stop at all, i.e., there was an SBE).

To mark the above ground truth, we manually went through the videos offline. We used this offline procedure instead of attempting to record the ground truth in real time, concurrently with the data collection, since the latter approach would have been tedious and error prone. We marked the video frame numbers corresponding to the start and end of each SBE.

Note that the SBEs include those at DBS as well as other locations such as traffic signals and any unplanned stops in congested traffic. From the set of SBEs, we manually identified the subset which corresponds to a DBS. To characterize where with respect to a DBS the SBE occurs, we have manually marked the video frame number corresponding to the last frame where the bus-stop is visible. Fig. 2 explains this. In this example, the bus stopped past the designated bus stop: $F_{SBE-start}$ and $F_{SBE-end}$ come after F_{DBS}.

B. Detecting Stop-Of-Bus Events (SBEs)

We first consider some simple alternatives and show their limitations, which then motivate our camera-based approach.

1) GPS-speed based approach: The first approach to detecting SBEs, we consider using the GPS sensor, which reports location as well as the estimated speed of movement. We can, in principle, apply a threshold on the speed estimate to infer if the vehicle (containing GPS unit), is stopped or moving.

To evaluate the effectiveness of this approach, we use the ground-truth marked data described above. For each video frame, we have marked its correspondence to a stopped state or a moving state. And for each such frame, we also have a GPS-reported speed estimate[1]. We divide our trace into stopped versus moving time periods, as per the marked ground truth.

Fig. 3 shows the cumulative frequency of the reported speed values: a plot each is shown for the stopped versus moving time periods. As is apparent from the graph, the natural threshold of $0m/s$ would result in both significant false positives (moving state detected as stopped: $\simeq 15\%$) as well as significant false negatives (stopped state detected as moving: $\simeq 47\%$). Even a threshold as high as $2m/s$ would result in over 30% false negatives and over 20% false positives. These results are in consonance with the GPS real-time speed estimation error of 3.1*mph* reported in SenSpeed [21].

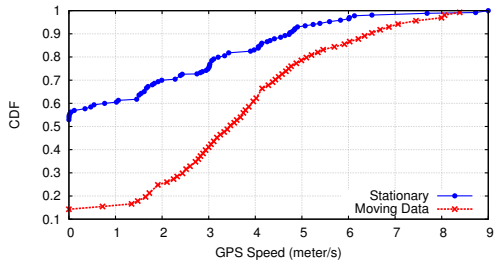

Fig. 3. CDF of GPS speeds

Thus it is clear that use of GPS sensor is erroneous for our purpose of distinguishing a rolling stop from a proper stop.

2) Accelerometer-based approach: The next possibility we consider to detect SBEs is using the accelerometer. Intuitively, we expect that when the bus is stopped, there would be less jerkiness and vibrations than when the bus is moving. To build an accelerometer-based vehicle state classifier, we considered time windows of 1-second. Note that a significantly larger time window is not suitable in our context, as an SBE may not last more than a few sec, even at DBSs [3]. Each 1-second window is marked with the ground truth: stopped versus moving.

We were unable to find prior work that uses accelerometer for our specific problem of detecting vehicle stoppings. But accelerometer usage has been explored for activity recognition in the past [22], [23]. These works use time domain features such as mean, standard deviation and energy as well as frequency domain feature such as entropy. Along these lines, we tried an SVM (Support Vector Machine) based classifier with the following accelerometer-based features: mean, standard deviation, energy and entropy of the readings along the vehicle's direction of motion, and standard deviation along the vertical direction. This gives an FPR of 21.1% and an FNR of 30.4%. These values are little better than the use of GPS.

To dig deeper, we examine whether at all accelerometer data from the moving and stopped states can be distinguished as belonging to different distributions. For this, we employ a standard Kolmogorov-Smirnov (KS) test on pairs of 1-second worth of z-axis (vertical direction) data (at 20Hz). We performed 23 such KS tests on a particular bus run with 23 SBEs. The null hypothesis was *not* rejected at the 95% confidence level, 54% of the time. That is, with the given accelerometer data, it was not possible to distinguish stopped versus moving states. This observation matches the earlier observation with the SVM classifier (which only used the second moment, i.e. the standard deviation)[2].

[1]Android API for GPS is a triggered update API. For a particular video frame, we simply take last updated value of GPS speed as the speed estimate.

[2]We note however that both the KS test and the SVM features consider only aggregate statistics and not the temporal pattern of the signal.

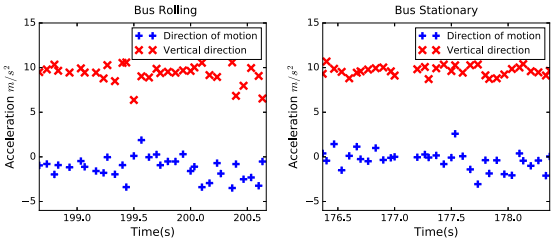

Fig. 4. Accelerometer: Moving vs Stationary

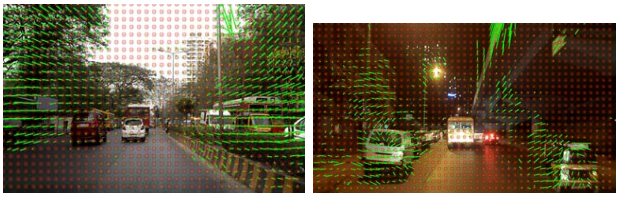

Fig. 5. Radially outward optical flow: day, night

Fig. 4 plots the accelerometer readings along the two axes for representative 2-second windows in the moving and stationary states. We can visually observe what is shown in the KS tests: the two states are not that different.

To explain this, we examined our video data closely. We found that false negatives are primarily due to vibrations, which happen even in stopped state, say because of the engine in an old and rickety bus or the jostling of a crowd of passengers climbing up or down steps to board or alight from the (high-floor) bus. Conversely, we noticed that false positives arise when there is very little vibration (on smooth roads) even as the bus rolls into a stop or starts moving from a rest position.

Thus, in sum, distinguishing between stopped and moving states based on the amount of vibrations, as measured by the accelerometer is error-prone too. We now turn to the design of the camera-sensor based classification approach. In our evaluation in Sec. IV-A, we also consider combining the camera-based classification with the accelerometer sensor, but this has marginal benefits.

C. S1: SBE Detection

As we explored the various algorithm designs for the video-based SBE detection, we used a subset of the road videos for the analysis. This helped us in quickly exploring the space of design options and converging on the effective choices. We used a subset of 6 videos for this purpose.

The key technique that FullStop uses is that of computing optical flow from images. Given two images, optical flow calculates how the pixels or other features "flow" from one image to the other. Flow vectors are used commonly in video analysis to detect motion [24]. When the vehicle moves forward with respect to the surrounding environment, we expect to see flow vectors computed across successive frames pointing radially outward. This is shown in Fig. 5.

Based on this intuition, we define a metric we term as the Radial Flow Vector (RFV) metric, which just the sum of the flow vectors' radial components.

TABLE II
PRELIMINARY RFV ANALYSIS

Metric	FPR(%)	FNR(%)
RFV_{pos}	9.8	24.0
RFV_{pos+lh}	9.6	20.9
$RFV_{pos+lhbot}$	11.9	14.5

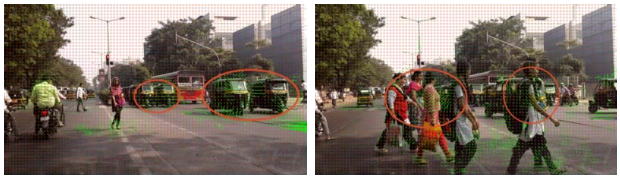

Fig. 6. Optical flow due to: oncoming traffic, people

$$RFV_{all} = \sum_i F\vec{V}_i.\vec{r_i} \quad (1)$$

Here FV_i is the i'th flow vector across a pair of frames, and r_i is the unit outward radial vector passing through the tail of FV_i, and . denotes the dot-product of vectors.

The RFV metric we use as a feature for the classification of a time-window as being in stopped or moving state, is the average of the metric values computed across each successive pair of frames in the time-window. For instance, we use a frame rate of $10fps$, then the RFV metric for a 1-second time window will be the average of RFV metrics computed across the 9 pairs of successive frames in that 1-second window.

There is a practical issue with the above metric, arising when the bus is stationary. As a bus is stopped, several vehicles could go past the bus, and these contribute flow vectors with negative radial components (i.e. radially inward). Note that any negative radial component flow vector could be caused only due to overtaking/passing vehicles, or due to reverse motion of the bus itself. Since the latter scenario is quite rare, we modify the RFV metric as:

$$RFV_{pos} = \sum_i pos(F\vec{V}_i.\vec{r_i}) \quad (2)$$

where $pos(x) = x$ if $x > 0$, 0 otherwise. That is, we take into account only those flow vectors which have a positive radial component.

Table II summarizes the FPR and FNR of the various design options we considered. The first row shows the results for the RFV_{pos} metric. We can see that the false negative rate is quite high. On closer examination of several instances of false negatives, we uncovered another practical issue with the RFV metric, not addressed in RFV_{pos}. This issue too arises when the bus is stationary. There are flow vectors with positive radial components (i.e., radially outward) caused due to oncoming traffic. After all, a stationary bus has forward relative motion with respect to oncoming traffic (example shown in Fig. 6(a)).

We realized that this is a problem especially on relatively narrow roads with no central road divider or only a small divider. To address this problem, we apply the intuition that such 'spurious' flow vectors are mostly toward the right side

TABLE III
EFFECT OF VIDEO RES. & FRAME RATE

FPR(%)	FNR(%)	Resolution	FPS
11.9	14.5	1920 x 1080	30
16.7	17.2	960 x 540	30
13.2	5.4	960 x 540	5
10.7	6.0	960 x 540	3
19.2	9.2	960 x 540	1

of the video frame (where the opposing traffic appears). So we further refine the RFV metric as:

$$RFV_{pos+lh} = \sum_{i \in LH} pos(\vec{FV_i}.\vec{r_i}) \quad (3)$$

where LH is the set of indices of flow vectors in the left half of the frame. This takes into account only those flow vectors which have a positive radial component, and whose tails are in the left half of the frame.

The second row of Table II shows the FPR and FNR for the RFV_{pos+lh} metric. While the FNR has decreased, it is still quite high. Once again, we took a close look at the video frame instances which contributed to such false negatives. This led to us uncovering another contributing factor to false negatives: people movement right in front of the vehicle, when it is stopped at a traffic signal. We realized that the magnitude of the flow vectors contributing to such false negatives was very high. An example of this is shown in Fig. 6(b). To address this source of error, we further tune the RFV metric as:

$$RFV_{pos+lhbot} = \sum_{i \in LHBOT} pos(\vec{FV_i}.\vec{r_i}) \quad (4)$$

Here $LHBOT$ is a subset of LH corresponding to the bottom 20% by magnitude. That is, we discard the top 80% of the flow vectors by magnitude. The intuition here is that if a vehicle is in motion, we expect *all* flow vectors to have a positive radial component, including the bottom 20%. This is akin to the human brain deducing that the vehicle in which one is sitting, is moving, even if there are only a few things in the background which appear to be moving. Here the percentile number 20 is chosen empirically: we found that our algorithm is not very sensitive to the exact value so long as we discard the top vectors by magnitude as outliers.

The third row of Table II shows the FPR and FNR results for $RFV_{pos+lhbot}$. The FNR has significantly reduced in comparison with RFV_{pos+lh}. We use this as our final RFV metric: henceforth, RFV will refer to $RFV_{pos+lhbot}$.

Video resolution, frame rate: How does the choice of video resolution affect the classification accuracy, and how does it affect the computational requirement? Thus far we have used a video resolution of 1920x1080 at a frame rate of 30 fps. Table III shows the FPR and FNR in classification for different choices of video frame rate and resolution.

The main computational step during real-time classification is the flow-vector computation; subsequent to that, we simply have to apply a pre-learnt threshold on the $RFV_{pos+lhbot}$ metric for the actual classification. The flow-vector computation is, in turn, linear in the number of pixels. Thus the computation required in each 1-second window is $T_{comp} \propto F \times N_{pixels}$, where F is the number of frames per second.

Fig. 7. Left lane divider vs Right lane divider

We see in Table III that the classification accuracy decreases with decrease in frame resolution. However, there is a significant improvement in the classification accuracy with decrease in frame rate. This is due to the fact that the flow vectors computed across frames which are further apart in time are less noisy as compared to when they are close together in time. We see that the best accuracy is achieved at a frame rate of $3fps$. Reasonable FPR and FNR values are achieved for a frame size of 960×540 (50% resolution in our camera) at this frame rate. We thus use these parameter values in our RFV based classification algorithm.

D. S2: Detecting Lateral Position ($LatPos_{SBE}$)

Bus stopping safely at the left edge of the road, and stopping unsafely one lane over, might only mean a lateral displacement too small to be detected reliably with a GPS fix. So just as in the case of S1 above, FullStop turns to using the camera. More concretely, we wish to detect cases where the bus has stopped in a lane which is not the left-most. For ease of explanation, we shall consider a scenario where there are two lanes in the direction of travel, and flag the stops on the right-lane as "unsafe" – laterally away from the bus-stop.

A natural option to consider for the above detection is the presence of lane markers and/or central divider.

The feature related to the central divider is based on the intuition that the straight line in the image, corresponding to the divider appears "different" when the bus is in the left lane versus the right lane. This is shown in Fig. 7. The difference is captured in terms of (a) θ_x, the angle the line segment makes with the x-axis, and (b) y_{rt}, the y-coordinate of the intersection with respect to the right edge of the image frame.

To compute the above two metrics, we first detect all the line segments in the image. We filter out lines whose $\theta_x \leq 0^\circ$ or $\theta_x \geq 90^\circ$ – cases where the central divider would make such angles are implausible and such line segments likely correspond to other features (e.g. a lamp post). We then identify L_{med}, the line segment with the median θ_x in the above set; the feature for that particular image frame is then the tuple (θ_x, y_{rt}) of L_{med}.

A problem with the above feature is that lane markers are absent on most roads in developing countries, except in highways/freeways. Central dividers (separating the two directions of traffic on a road) are more common, but these too are not universal on city roads.

We therefore use another feature. This is based on the intuition that when a bus is stopped on the right lane, other vehicles generally overtake/pass the bus from its left side – in fact, this is precisely the reason why such stops are unsafe. To capture this, we compute a flow vector metric comparing a given frame to its previous one. This flow vector is computed using only the left half of the frame as we are only interested in vehicles overtaking from the left side. We take the average magnitude of those flow vectors which have a negative y-component and a positive x-component (i.e. corresponds to overtaking vehicles). We call this feature FV_{left}.

The above three metrics $\theta_x, y_{rt}, FV_{left}$ are all averaged over the frames corresponding to a 1-second duration starting from the detection of the SBE: $\bar{\theta_x}, \bar{y_{rt}}, \bar{FV_{left}}$. These averages are then used to train a machine learning classifier.

Note that unlike the case of $LongPos_{SBE}$ detection where we did not rely upon manual labels, here we have to use manual labels in the training process. Hence, while for $LongPos_{SBE}$ determination, we could afford a per-bus-stop training, here we need a model which can work across roads. We shall show in our evaluation that this is indeed possible.

IV. Evaluation of FullStop

We now present an evaluation of the two mechanisms in FullStop using data from roads of Mumbai. It is to be noted that algorithms for detecting S1, S2 need to be run only in the vicinity of DBSs. We may use GPS, or any other approximate location mechanism to determine that the bus is in the vicinity of a DBS and only then invoke the camera-based analysis.

A. Evaluating SBE Detection

This section presents the results of evaluation of the camera-based mechanism to identify SBEs. We use the data collected on the roads of Mumbai, as summarized in Table I. The following results are presented for a frame rate of $3fps$ and an image resolution of 960 x 540 (taking only the left half of which will give 480 x 540).

Table IV summarizes the FPR and FNR metrics for the evaluation of the RFV metric based identification of SBEs. Rows 1-3 of the table present the results of the standard 10-fold cross-validation, with SVM (Support Vector Machine) as the underlying algorithm. We can see that performance is significantly better than the alternatives based on GPS or accelerometer (Sec. III-B): FPR and FNR are both under 11%.

In the first two rows, we have used day-time training for classification in day-time, and likewise for night-time, respectively. The third row presents the evaluation where we do not distinguish between day versus night. We initially expected that separate training for daytime versus nighttime would be required as we likely have much less visibility of finer features of the surrounding environment at night than during the day. Contrary to this expectation, we find that the same RFV based mechanism works well at night time too, with night-time data used for training (row 2 in Table IV), with classification accuracy being similar to that during daytime (row 1). Furthermore, the accuracy remains similar even when all data, from the daytime and nighttime, is mixed together (row 3). This is likely because even in night time, there are enough static features in the surrounding environment visible for the algorithm to be effective. Thus, no separate day versus night training is necessary.

TABLE IV
Evaluation of RFV based SBE detection

	Approach	Time	FPR(%)	FNR(%)
1	RFV	Day	11.0	6.3
2	RFV	Night	8.0	7.1
3	RFV	All	10.9	7.8
4	RFV+Acc	All	11.4	6.1
5	Train:RtA, Test:RtB	Day	15.4	10.3
6	Train:RtB, Test:RtA	Day	8.8	7.2
7	Train:RtA, Test:RtB	Night	3.7	23.8
8	Train:RtB, Test:RtA	Night	16.0	4.4

TABLE V
Data used for lateral positioning

	Left lane stopping	Right lane stopping
Route-A(day)	34 mins (9 stops)	64 mins (13 stops)
Route-B(day)	29 mins (7 stops)	72 mins (14 stops)
Route-B(night)	15 mins (4 stops)	35 mins (7 stops)

Combining camera and accelerometer sensors: Can the RFV metric be used in combination with the accelerometer data, to improve the overall classification accuracy, in a multi-modal sensing approach? To answer this question, we considered the accelerometer features along with the $RFV_{pos+lhbot}$ metric for SVM-based classification. Row 4 in Table IV shows the results. We see that the use of the accelerometer sensor neither significantly improves nor significantly degrades the FPR or FNR. We also examined whether a decision-tree based approach will result in an improved performance of the multi-modal approach, but the results for these were similar to the use of SVM (and hence not shown).

A disadvantage of the above approach is that we require manually labeled training data. Clearly, it would be infeasible to train and manually label video data for all roads. A relevant question then is, can one road's training data be used for classification on another? To evaluate this, we identify two different routes, A and B, in our dataset, and examine the effect of training on one and testing on another. Route A is relatively less congested and has better road conditions compared to route B. Rows 5-8 in Table IV present these results. We see that the FPR and FNR tend to increase in some cases but are still better than for GPS and accelerometer based detection.

In terms of the prevalence of rolling stops at designated bus stops (DBSs), in our dataset, we found (for a subset of 5 bus runs analyzed manually), we observed that 9 of the 55 total stops, i.e. nearly 1 in 6 stops were in fact unsafe rolling stops.

B. Evaluation of $LatPos_{SBE}$ Classification

We have evaluated our lateral position detection algorithm as follows. Our aim is to detect whether the bus has stopped in the correct lane. Although lateral displacement does happen (see videos at http://tinyurl.com/z4vt4c5), our bus data set has infrequent instances of this, thus making it unsuitable for

TABLE VI
LATERAL POSITIONING RESULTS

Features	FPR(%)	FNR(%)
Optical flow	14.2%	23.8
Central divider	8.6%	20.7
All	5.4%	10.2

TABLE VII
MAXIMUM SUPPORTABLE FPS

Workload	Max fps
Nil	32.36
No Threading (SBE detection)	2.48
Multi-Threading (SBE detection)	4.84
Multi-Threading (SBE detection + $LatPos_{SBE}$ classification)	3.81

evaluation of our algorithm. So we emulate stop-of-bus in the right lane as follows. We hired a taxi and made it to stop on the right side of the road. This allows us to have better control over the experiments. Now since live traffic plies in the right lane, it was difficult to halt the taxi in the right lane for an extended period of time (of even a few seconds). Hence, for this, we chose a set of places where civil work was going on the right side of the road, and all the traffic is bound to travel on the left lane. We found several different such spots on the right side of the road and stopped the taxi at these spots. The summary statistics of the data collection is in Table V.

Table VI gives results of a decision-tree classifier using the features mentioned in Sec III-D. These results are for a standard 10-fold cross validation. We see that both the optical flow feature (FV_{left}) as well as the central divider related features ($\bar{\theta}_x, \bar{y}_{rt}$) are not individually as successful as their combination. The combination of the 2 sets of features gives an overall FPR & FNR of about 5% and 10% respectively.

V. FULLSTOP ANDROID APP

In order to test the efficacy of FullStop in real-time, we prototyped an android application (510 LoC, excluding libraries). We have used OpenCV library for vision related utilities. In the current version, the application has two modules: SBE detection and $LatPos_{SBE}$ classification. In our tests below, we have run the app on an LG Nexus 5x phone. Table VII lists the effective frames-per-second (fps) obtained for different scenarios. Optical flow calculation is an essential part of both the modules. We observed that the effective fps obtained is below 3 fps, if optical flow is calculated without using threads. Also, the CPU which is multi-core remains under utilized in such a case. We found that, if two worker threads operate upon two different parts of the frames, the effective fps obtained is above 4, which is acceptable considering the fact that we evaluated our algorithms for 3 fps (Sec. IV-A). Further, it is observed that on adding $LatPos_{SBE}$ classification module, which involes finding lines in a frame, effective fps drops below 4 but remains above 3. It is possible to improve upon this performance further, by using a GPU, if available; in our current prototype, although the LG-Nexus phone has a 1.8GHz hexa-core 64-bit Adreno GPU, the library we have used (OpenCV) does not support this. Improving our prototype along this dimension is part of our future work.

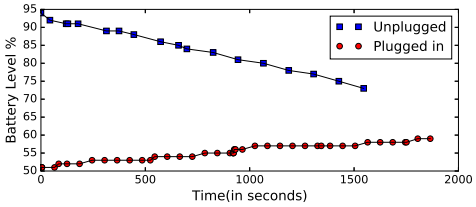

Fig. 8. Battery level as a function of time

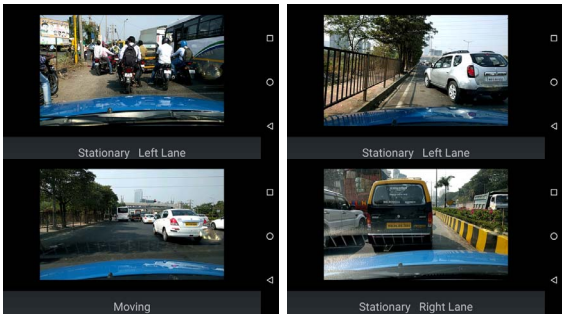

Fig. 9. FullStop app: SBE detection, $LatPos_{SBE}$ classification

Along with running optical flow, the application also potentially needs to keep the GPS activated, to implement location triggered camera operation, i.e. turn on camera only in the vicinity of bus stops. When camera and GPS can be ON, is the app sustainable in terms of energy consumption? In our test runs on the road, we observed a near 20% fall in battery levels when the app is operated for around 25 minutes. However, when the phone is plugged-in using a 1.5A charger, it becomes self-sustainable. Fig. 8 shows the phone's energy levels with and without plugging-in.

The prototype FullStop application shows the current view of the camera as well as a status bar at the bottom which shows the decision made by the two modules. Some screenshots of the app are shown in Fig. 9. We also share the current version of the apk here [3].

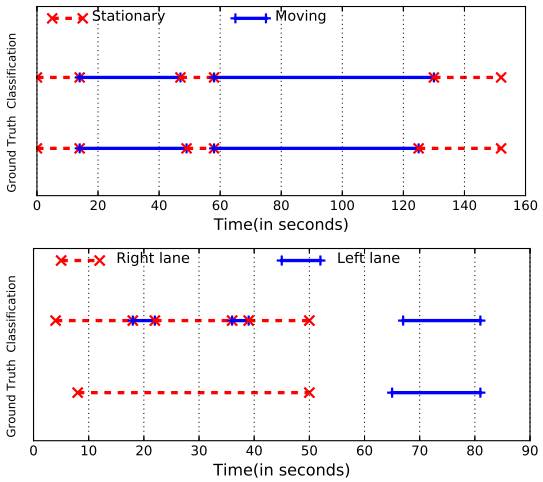

Fig. 10. Event detection: SBE, $LatPos_{SBE}$

Sample run of FullStop app on road:

To demonstrate the viability of the app under various real-road scenarios, but with vehicle movement under our control, we engaged a taxi and performed runs in day-time as well as night-time. Fig. 10(a) shows a sample marking of ground truth as well as the SBE detection algorithm for an illustrative time-window. The vehicle is stationary in the window 0-14 seconds, which is classified correctly by the app. Misclassification (false positives) occurs in the interval 47-49 seconds, where vehicle is classified stationary even though it is moving. This occurs because during this time interval, another vehicle occupied the whole view of the camera resulting in low RFV metric value. Correct classifications occur in intervals 49-58 seconds when vehicle was stationary and 58-125 seconds when it is moving. Misclassification (false negatives) occurs again in the window 125-130 seconds, where vehicle is classified moving as opposed to stationary. This occurs due to people crossing over from right to left, very close to the vehicle, hence resulting in high RFV value. However, false classifications are overall rare, with overall accuracy of over 90% in day as well as night. The overall FPR was 2% and FNR was 9.6%; these values are similar to that reported in Table IV in Section IV.

The overall classification accuracy for $LatPos_{SBE}$ was 81% in our test run. Fig. 10(b) shows a plot for $LatPos_{SBE}$ classification for an illustrative time window of our test run. In this plot, unmarked stretches of time represent the state when the vehicle was moving. For instance, in this plot, the vehicle was moving 0-8 seconds after which it became stationary in the window 8-50 seconds. Recall here that the $LatPos_{SBE}$ detector is triggered only after a vehicle is classified as stationary. False SBE was detected at 4 seconds (vehicle at very low speed), which in turn triggered $LatPos_{SBE}$. Since the vehicle was in the right lane, it was classified so.

This test run also revealed some situations in which the $LatPos_{SBE}$ classification algorithm can be erroneous. In heavily congested traffic, there is significant occlusion, due to which we are unable to spot any lane dividers, and nor are there (m)any optical flow vectors. In such situations, our current algorithm has no feature to rely upon for classification. Such situations arose in intervals 18-22 seconds and 36-39 seconds in Fig. 10(b). Misclassifications also occurred between 65-67 seconds where vehicle was in the left lane but classified as in right lane. This too happened in congested traffic, but due to straight lines on other vehicles to the right being confused with the road divider. We are in the process of exploring the use of other sensors such as the gyroscope to detect lane changes [13], in combination with our current set of features, to handle such scenarios of heavy optical occlusion.

VI. Conclusion

This paper has focused on an important aspect of road safety: the stopping behaviour of buses. Using the camera sensor of a windshield-mounted smart-phone, and a set of optical flow based metrics, we have carefully designed algorithms in our system, FullStop, to detect two unsafe behaviours: rolling stops at designated bus stops (DBSs), and lateral displacement from DBSs. We have evaluated the algorithms of FullStop on an extensive set of data from city roads and shown that these are effective in achieving low false positive and false negative rates. We have also built a prototype Android app of FullStop to show the practicality of our approach.

Acknowledgment: The author, Ravi Bhandari would like to thank Microsoft Research and Confederation of Indian Industry for the financial support provided under Prime Minister's fellowship for doctoral research. We thank Laxman for his support in collecting video data and Sridhar for the ground truth marking. The authors are also grateful to the anonymous COMSNETS reviewers for their insightful feedback.

References

[1] *List of countries by traffic-related death rate*, https://en.wikipedia.org/wiki/List_of_countries_by_traffic-related_death_rate.
[2] *DTC's killer buses roam free on Delhi roads*, Jan 2014, http://www.cmsindia.org/mediacoverage/Jan2014_DTC_buses_highlight.pdf.
[3] *Sample videos*, http://tinyurl.com/z4vt4c5.
[4] *Internal Technical Report*, https://www.cse.iitb.ac.in/ ravib/fullstop-tech-report.pdf.
[5] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones," in *ACM SenSys*, 2008.
[6] 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 *ACM MobiSys*, 2008.
[7] P. Zhou, Y. Zheng, and M. Li, "How long to wait?: Predicting bus arrival time with mobile phone based participatory sensing," in *ACM MobiSys*. ACM, 2012, pp. 379–392.
[8] E. Koukoumidis, L.-S. Peh, and M. Martonosi, "SignalGuru: Leveraging Mobile Phones for Collaborative Traffic Signal Schedule Advisory," in *ACM MobiSys*, 2011.
[9] *iOnRoad*, http://www.ionroad.com/.
[10] R. Sen, A. Cross, V. N. Padmanabhan, E. Cutrell, and B. Thies, "Accurate Speed and Density Measurement for Road Traffic in India ," in *ACM DEV*, 2013.
[11] *Google Self-Driving Car Project*, https://www.google.com/selfdrivingcar/.
[12] H. Hartenstein and L. Laberteaux, "A tutorial survey on vehicular ad hoc networks," *IEEE Communications magazine*, vol. 46, no. 6, pp. 164–171, 2008.
[13] D. Chen, K.-T. Cho, S. Han, Z. Jin, and K. G. Shin, "Invisible Sensing of Vehicle Steering with Smartphones," in *ACM MobiSys*, 2015.
[14] C.-W. You, N. D. Lane, F. Chen, R. Wang, Z. Chen, T. J. Bao, Y. Cheng, L. T. Mu Lin, and A. T. Campbell, "CarSafe App: Alerting Drowsy and Distracted Drivers using Dual Cameras on Smartphones," in *ACM MobiSys*, 2013.
[15] P. Handel, I. Skog, J. Wahlstrom, and F. Bonawiede, "Insurance Telematics : Opportunities and Challenges with the Smartphone Solution," vol. 6, no. 4, pp. 57–70, 2014.
[16] J. Wahlstrom, I. Skog, and P. Handel, "Driving Behavior Analysis for Smartphone-based Insurance Telematics," in *Workshop on Physical Analytics*, 2015.
[17] *Cambridge Mobile Telematics*, http://www.cmtelematics.com/.
[18] *Zendrive*, https://www.zendrive.com/.
[19] *LightMetrics*, http://www.lightmetrics.co/.
[20] *Raksha SafeDrive*, https://www.raksha.me/.
[21] J. Yu, H. Zhu, H. Han, Y. J. Chen, J. Yang, Y. Zhu, Z. Chen, G. Xue, and M. Li, "Senspeed: Sensing driving conditions to estimate vehicle speed in urban environments," *IEEE Transactions on Mobile Computing*, vol. 15, no. 1, pp. 202–216, 2016.
[22] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," in *Aaai*, vol. 5, no. 2005, 2005, pp. 1541–1546.
[23] X. Su, H. Tong, and P. Ji, "Activity recognition with smartphone sensors," *Tsinghua Science and Technology*, vol. 19, no. 3, pp. 235–249, 2014.
[24] J. H. Duncan and T.-C. Chou, "On the detection of motion and the computation of optical flow," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 3, pp. 346–352, 1992.