Leveraging Smartphone Cameras for Collaborative Road Advisories

Emmanouil Koukoumidis, *Member, IEEE,* Margaret Martonosi, *Fellow, IEEE,* and Li-Shiuan Peh, *Member, IEEE*

Abstract—Ubiquitous smartphones are increasingly becoming the dominant platform for collaborative sensing. Smartphones, with their ever richer set of sensors, are being used to enable collaborative driver-assistance services like traffic advisory and road condition monitoring. To enable such services, the smartphones' GPS, accelerometer and gyro sensors have been widely used. On the contrary, smartphone cameras, despite being very powerful sensors, have largely been neglected. In this paper, we introduce a collaborative sensing platform that exploits the cameras of windshield-mounted smartphones.
To demonstrate the potential of this platform, we propose several services that it can support, and prototype SignalGuru, a novel service that leverages windshield-mounted smartphones and their cameras to collaboratively detect and predict the schedule of traffic signals, enabling Green Light Optimal Speed Advisory (GLOSA) and other novel applications. Results from two deployments of SignalGuru, using iPhones in cars in Cambridge (MA, USA) and Singapore, show that traffic signal schedules can be predicted accurately. On average, SignalGuru comes within 0.66s, for pre-timed traffic signals and within 2.45s, for traffic-adaptive traffic signals. Feeding SignalGuru's predicted traffic schedule to our GLOSA application, our vehicle fuel consumption measurements show savings of 20.3%, on average.

Index Terms—smartphone, camera, Intelligent Transportation Systems, services, traffic signal, detection, filtering, prediction, collaboration

1 Introduction

With an ever richer set of sensors, increased computational power and higher popularity, smartphones have become a major collaborative sensing platform. In particular, smartphones have been widely used to sense their environment and provide services to assist drivers. Several systems have been proposed that leverage smartphone GPS, accelerometer and gyroscope sensors to estimate traffic conditions [11], [21], detect road abnormalities [21] and compute fuel-efficient routes [8].

Cameras, in contrast to other smartphone sensors, have so far been underutilized for automated collaborative sensing. Cameras have been used only for a handful of participatory sensing systems; both image capture and image analysis are performed by a human user. Such applications include the monitoring of vegetation, garbage, and campus assets [24]. In all these services, users must point their smartphone camera to the target object, capture an image and upload it to the central service where a human operator will analyze it. The adoption of collaborative sensing services that leverage smartphone cameras without manual user and operator effort has so far been hindered because of two false beliefs: 1) the view of smartphone cameras is always obstructed (*e.g.,* carried in pockets or placed flat on the table), and 2) image processing requirements are prohibitively high for resource-constrained mobile devices.

- *Emmanouil Koukoumidis is with the Department of Electrical Engineering, Princeton University, NJ, 08540.*
- *Li-Shiuan Peh is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, MA, 02139.*
- *Margaret Martonosi is with the Department of Computer Science, Princeton University, NJ, 08540.*

In this paper, we propose a novel collaborative sensing platform that is based on the cameras of windshield-mounted smartphones. We show that accurate and near-real-time camera-based sensing is possible. Many drivers are already placing their phones on the windshield in order to use existing popular services like navigation. Once a phone is placed on the windshield, its camera faces the road ahead. Our proposed sensing platform leverages these cameras to opportunistically capture content-rich images of the road and the environment ahead. Inter-device collaboration is also leveraged to gather more visual road-resident information and distill it into knowledge (services) that can be provided to the drivers. With their cameras, a network of collaborating windshield-mounted smartphones can enable a rich set of novel services.

In this paper, we focus on the description and evaluation of the SignalGuru service [17]. SignalGuru leverages the cameras of windshield-mounted smartphones in order to detect traffic signals ahead and predict their future schedule. SignalGuru devices collaborate with other regional devices in order to mutually improve their historic traffic signal status information and better predict when the signal ahead will turn green/red.

Providing real-time services, like SignalGuru, on top of a confederation of windshield-mounted smartphones and their cameras poses several challenges that need to be overcome:

1) *Commodity cameras*: The quality of smartphone cameras is significantly lower than that of high-end specialized cameras used in computer vision and autonomous navigation. Smartphone cameras have both lower color quality and lower resolution. Further, as capturing still images is very slow (1-2 seconds on an iPhone 3GS device), video frames should often be used instead for low-overhead and high-frequency image-based detection. This further degrades resolution, as video resolution

is only up to 640x480 pixels for iPhone 3GS and 1280x720 pixels for iPhone 4 devices.

2) *Limited processing power*: Processing video frames to detect visual information takes significant computational resources. In SignalGuru, traffic signal detection is the most compute-intensive task. A traffic signal detection algorithm that runs on resource-constrained smartphones must be lightweight so that video frames can still be processed at high frequencies. The higher the processing frequency, the more accurately SignalGuru can measure the duration of traffic signal phases and the time of their status transitions.
3) *Uncontrolled environment composition and false detections*: Windshield-mounted smartphones capture the real world while moving. As a result, there is no control over the composition of the content captured by their video cameras. Results from one of our deployments suggest that the camera-based traffic signal detection algorithm can confuse various objects for traffic signals and falsely detect traffic signal colors. A misdetection rate of 4.5% can corrupt up to 100% of traffic signal predictions. Schemes need to be devised to carefully filter noisy image-based object detections.
4) *Variable ambient light conditions*: Still image and video capture are significantly affected by the amount of ambient light that depends on both the time of the day and the prevailing weather conditions. By adjusting the camera exposure time to the fixed luminous intensity of traffic signals, SignalGuru robustly detects traffic signals regardless of the prevailing ambient light conditions.
5) *Need for collaboration*: The visual information that an individual smartphone senses is limited to its camera's view angle. Regional smartphones thus need to collaborate in order to increase their information reach. In the SignalGuru service, for example, a device may not be able to see a far-away traffic signal, or may not be within view of the traffic signal for a long enough stretch of time. Collaboration is needed between vehicles in the vicinity (even those on intersecting roads) so that devices have enough information to be able to predict the schedule of traffic signals. Collaboration is also needed in order to maintain SignalGuru's data over time and in a distributed fashion within the vehicular network.
   Alternatively, collaborative services like SignalGuru could be implemented on an Internet server, relying on always-available long-range cellular communication to the server. However, in this paper we focus on a completely infrastructure-less solution that relies solely upon opportunistic short range communication (ad-hoc 802.11g) among the windshield-mounted devices. Such a grassroots service architecture may have a more complex design, but avoids the cost of internet servers and the costly, high-latency, low-bandwidth and potentially overloaded long range cellular connections [15], [16].

It should be noted that we do not consider battery lifetime as a major challenge. Mobile phones can be plugged into the ample energy resources of a vehicle. In cases where this does not hold, approaches proposed for lifetime maximization in sensor networks [13], [32] can be used. Such approaches can determine if and when a given device needs to perform certain power hungry-tasks (*e.g.,* SignalGuru traffic signal detection, collaboration with wireless communication).

The contributions of this work are the following:

1) We show that, with their cameras, collaborating windshield-mounted smartphones create a powerful sensing platform enabling a rich set of novel services. We discuss five such services and prototype SignalGuru demonstrating the ability of windshield-mounted smartphones to effectively detect and predict the schedule of traffic signals. Not only pre-timed but also state-of-the-art traffic-adaptive traffic signals can be predicted with very good accuracy (2.45s) by using customized Support Vector Regression (SVR) models.
2) Networks of windshield-mounted smartphones can greatly increase their camera-based sensing frequency and accuracy by fusing information from the smartphone's Inertial Measurement Unit (IMU) to reduce the video area that needs processing. Our IMU-based detection window, halves both the processing time and the misdetection rate for SignalGuru. We also propose and evaluate low-pass filtering and a colocation filter that effectively filter away false positive event (*e.g.,* traffic signal transition) detections.
3) Many user-focused applications can be built on top of the traffic signal prediction system aka SignalGuru. In particular, our GLOSA system offers speed advisories to avoid undue stops and waits at red lights. Testing this system using an onboard fuel efficiency monitor, we show that when drivers follow the advisory of our SignalGuru-based GLOSA system, 20.3% fuel savings can be achieved.

In the next sections, we describe SignalGuru in detail. In Section 2, we present the motivation behind SignalGuru and in Section 3 the SignalGuru-enabled GLOSA application. Section 4 describes the operation of traffic signals and Section 5 the architecture of our collaborative SignalGuru service. In Section 6, we present our experimental methodology and in Section 7, we evaluate the performance of SignalGuru's individual modules based on our two real-world deployments. Section 8 discusses the operation of SignalGuru in complex intersections. In Section 9, we describe four more services that windshield-mounted smartphones could support and discuss their challenges as compared to SignalGuru. Finally, Section 10 surveys related work and Section 11 offers our conclusions.

2 SIGNALGURU MOTIVATION

There are more than 272,000 traffic signals in major intersections of the USA alone [14], and our daily driving experience is significantly influenced by them. Traffic signals are widespread in developed countries as they allow competing flows of traffic to safely cross busy intersections. Traffic signals, however, do take their toll. The stop-and-go movement pattern that they impose, increases fuel consumption by 17%

[2], CO_2 emissions by 15% [2], causes congestion [3], and leads to increased driver frustration [14].

Drivers can be assisted with a Green Light Optimal Speed Advisory (GLOSA) system [2], [30]. A GLOSA system advises drivers on the optimal speed they should maintain when heading towards a signalized intersection. If drivers maintain this speed, the traffic signal will be green when they reach the intersection, allowing the driver to cruise through. In this way, the stop-and-go movement pattern can be alleviated.

Worldwide, only a handful of GLOSA systems have been deployed [30], and they have so far been based on roadside message signs (wired to traffic signals). These signs are placed a couple hundred meters away from the signal and display the optimal speed drivers should maintain. Their costly and often impractical deployment and maintenance, however, has hindered their widespread usage.

Countdown timers at vehicular traffic signals constitute another alternative approach to assist drivers; digital timers next to the traffic signal display the time till the signal changes from red to green and vice versa. Such traffic signals are deployed only in a few cities around the world. The cost of updating existing traffic signals to include such timers has hindered their widespread deployment.

Countdown timers for pedestrian traffic signals are much more common in the USA and the rest of the world, and drivers can sometimes use these to infer when the light will turn green. However, these are very often not visible from far away but only after one has reached the intersection. At that time, it is too late for drivers to adapt speed and so they need to come to a complete halt anyway. Furthermore, at some intersections, it is not easy or even possible for the driver to infer the time the signal will switch; the intersection may have a complex phase schedule and the green light for the driver may not come immediately after some pedestrian timer counts down to zero.

US and European transportation agencies recognize the importance of GLOSA and access to traffic signal schedules, and thus have advocated for the integration of short-range (DSRC) antennas into traffic signals as part of their long-term vision. DSRC-enabled traffic signals will be able to broadcast in a timely fashion their schedule to DSRC-enabled vehicles that are in range. Audi recently prototyped a small-scale DSRC-based GLOSA system for 25 traffic signals in Ingolstadt (Germany) [2]. Once again, however, the widespread deployment of such an approach has been hindered by the significant cost to equip traffic signals and vehicles with the necessary specialized computational and wireless communications infrastructure.

In this paper, we take an *infrastructure-less* approach to accessing traffic signal schedules by leveraging the proposed collaborative platform of windshield-mounted smartphones and their cameras. Windshield-mounted smartphones use their cameras to detect and determine the current status of traffic signals. Multiple phones in the vicinity use opportunistic ad-hoc communications to collaboratively learn the timing patterns of traffic signals and predict their schedule. SignalGuru's predicted traffic signal schedule then enables GLOSA and other possible applications on the phone.

3 SignalGuru Applications: GLOSA

The goal of the GLOSA application is to advise drivers on the optimal speed they should maintain so that the signal is green when they arrive at the next intersection. In this way the driver can cruise through the intersection without stopping. A GLOSA application can offer several benefits such as 1) decreased fuel consumption, 2) smoothed and increased traffic flow (stop-and-go patterns avoided), and as a result of these, 3) decreased environmental impact.

A GLOSA application needs four pieces of information in order to be able to calculate the optimal speed: 1) the residual amount of time till the traffic signal ahead turns green, 2) the intersection's (stop line) location, 3) the vehicle's current location, and 4) the queue length of the traffic signal ahead. The first is provided by SignalGuru, the second by map information [23] and the third by the available localization mechanisms on the mobile device (*e.g.,* GPS). The traffic signal queue length and the time it will take to discharge can be estimated by fusing information about the number and positions of vehicles in the queue as described in [6], [14].

If no traffic signal queue length information is available, and when vehicles are very close (<100m) to the intersection, GLOSA should switch from a speed advisory to a time countdown. Drivers can then look at the queue length ahead and manually estimate their optimal speed.

Although GLOSA may often advise a vehicle to reduce its speed, the vehicle's total travel time will not be increased. On the contrary, GLOSA helps decrease average travel time by 1%-18% [1]. Despite the speed reduction, a GLOSA-enabled vehicle will still travel through the intersection at the same traffic signal phase as it would if it were traveling at its regular speed. Moreover, at the time the signal turns green, a GLOSA-enabled vehicle will be cruising through the intersection with an initial non-zero speed, as opposed to a regular vehicle that would have to start from a complete halt.

GLOSA also improves the overall flow reducing congestion. The traffic flow is smoother and faster when vehicles are cruising through the intersections as opposed to when they come to a complete halt and then slowly accelerate one after the other to cross the intersection. Traffic flow improvements then lead to further gas and travel time savings.

The larger the available lead-up time *i.e.,* the amount of time in advance that predictions are available, the more effective GLOSA is. Predictions that are available at least 20 sec in advance, while the driver is perhaps 250m from the traffic light, provide enough room to control the vehicles' speed. The prediction accuracy should be less than 10% of the traffic signal's phase length to avoid wasting precious green time (*i.e.,* to avoid vehicles arriving at the intersection long after the light has switched to green).

Besides GLOSA, SignalGuru's traffic signal schedule predictions can be used to enable more applications like a Traffic Signal-Adaptive Navigation (TSAN) service and a Red Light Duration Advisory (RLDA) service [17]. A TSAN service will optimize the suggested route by also taking traffic signal schedules into account. An RLDA service would advise the drivers when switching off their engine would save them gas while waiting at a red light.

4 Traffic Signal Background

In signalized intersections, different but non-conflicting (safe to co-exist) vehicular and pedestrian movements are grouped together to run at the same time. Such groups of movements are termed phases. A simple intersection typically has two phases. When the light is green for phase A, vehicles or pedestrians moving North-South can safely move at the same time. Later the traffic signal will turn red for phase A and green for phase B. At this time, vehicles and pedestrians moving East-West can go. When this phase completes, the intersection has completed one *cycle* and the light will turn red again for phase B and green for phase A. Many intersections may have more than two phases. The amount of time that the light stays green in a given phase is *phase length*. The sum of all phase lengths of an intersection is *cycle length*.

Most traffic signals in the US are pre-timed traffic signals [25]. For pre-timed traffic signals the settings (phase lengths, cycle length) of the traffic signals are fixed and the exact same schedule repeats in every cycle. The settings only change when the intersection switches mode of operation depending on the day or the time of day. Typically pre-timed traffic signals have three modes of operation: 1) off-peak, 2) a.m. peak and 3) p.m. peak. Sometimes, there is a special schedule for Saturday peak.

In contrast to the US, Singapore adaptively controls its traffic signals using the state-of-the-art GLIDE system that is adapted from the SCATS system [27]. SCATS adaptively adjusts settings of the traffic signals based on measurements from its inductive loop detectors. One loop detector is installed per lane and placed beneath the road surface at the intersection stop line. Loop detectors, while the light is green, measure the saturation of their lane. Specifically, lane saturation is calculated as a function of the number of vehicles that traveled over the corresponding loop detector and the measured total gap time (*i.e.,* amount of time that the loop detector is unoccupied). Lane saturations are merged to calculate a phase's saturation.

SCATS adjusts traffic signal settings in order to balance the saturation across the different phases of the intersection. The higher the saturation of a phase (more vehicles), the greater portion of the cycle length is allocated to the specific phase. Cycle length duration is adjusted depending on the saturation of all the phases of the intersection and increases when the maximum phase saturation increases. Longer cycles allow intersections to operate more efficiently (higher throughput), but increase the waiting times and frustration of drivers. SCATS measures phase saturations and changes the intersection traffic signal settings accordingly every cycle, *i.e.,* every 1-3 minutes.

5 SignalGuru Architecture

SignalGuru aims to detect and predict the schedule of traffic signals using just software on commodity smartphones. It is a grassroots software service that leverages opportunistic sensing on mobile phones to detect the current color of traffic signals, share with nearby mobile phones to collectively derive traffic signal history, and predict the future status and timing of traffic signals.

Figure 1 shows the modules in the SignalGuru service. First, phone cameras are used to capture video frames, and detect

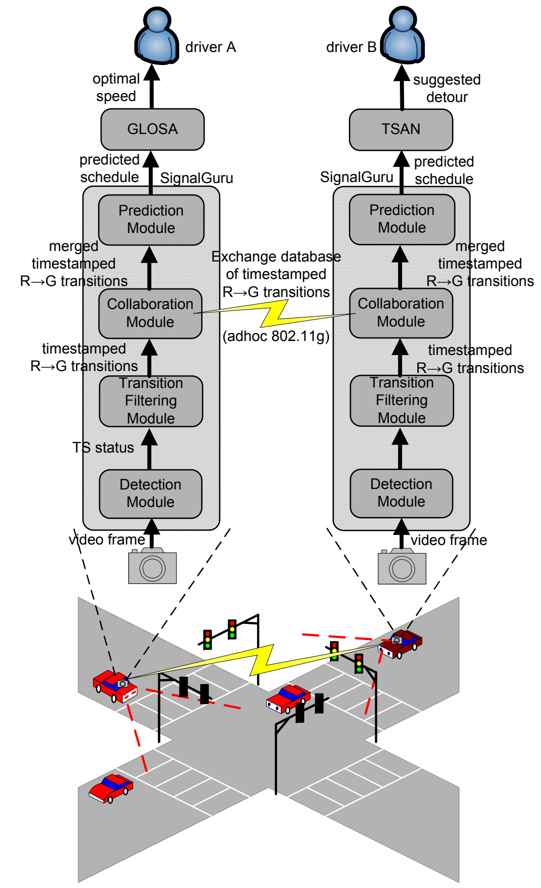


Fig. 1: **SignalGuru service architecture.**

the color of the traffic signal (detection module). Then, information from multiple frames is used to filter away erroneous traffic signal transitions (transition filtering module). Third, nodes running the SignalGuru service broadcast and merge their traffic signal transitions with others in communications range (collaboration module). Finally, the merged transitions database is used to predict the future schedule of the traffic signals ahead (prediction module).

The prediction of the future schedule of traffic signals is based on information about past timestamped R→G (red to green) transitions. The prediction is based on R→G transitions, as opposed to G→Y (green to yellow) transitions, because vehicle-mounted smartphones can witness and detect R→G transitions much more frequently; when the traffic signal is red, vehicles have to stop and wait till the signal turns green. As a result, it is quite likely that a vehicle will be at the intersection at the moment that the R→G transition occurs and thus detect it. For a G→Y transition to be detected, the vehicle needs to be driving towards the intersection and have good view (≤ 50 meters away) of the signal when the signal color changes. As a result, it is much less likely[1] for a vehicle to be close enough to the intersection at the moment of the G→Y transition. The same applies also for Y→R transitions. Section 5.4 discusses how timestamped R→G transition information is used to predict the traffic signal schedule.

1. In our Singapore deployment (Section 6.2), vehicles witnessed a total of 37 R→G transitions but only two G→Y transitions.

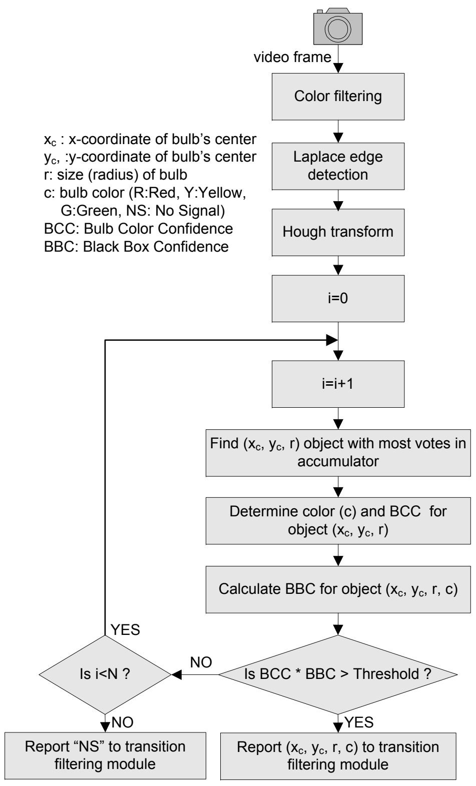

video frame
Color filtering
Laplace edge detection
Hough transform
i=0
i=i+1
Find (xc, yc, r) object with most votes in accumulator
Determine color (c) and BCC for object (xc, yc, r)
Calculate BBC for object (xc, yc, r, c)
Is BCC * BBC > Threshold ?
NO
Is i<N ?
YES
NO
YES
Report "NS" to transition filtering module
Report (xc, yc, r, c) to transition filtering module
xc : x-coordinate of bulb's center
yc, :y-coordinate of bulb's center
r: size (radius) of bulb
c: bulb color (R:Red, Y:Yellow, G:Green, NS: No Signal)
BCC: Bulb Color Confidence
BBC: Black Box Confidence

Fig. 2: **Traffic signal detection algorithm. "NS" stands for "No Signal" and is the status returned by the detection module when no traffic signal can be detected with a confidence higher than the threshold value.**

5.1 Detection Module

The detection module detects and reports the current color of potential traffic signals in the captured video frames. The detection module is activated based on its GPS location[2] and only when it is close (<50m) to a signalized intersection. The video frames are captured using the standard iPhone camera. As Figure 3 shows, when the smartphone is mounted on the windshield, this camera is facing outside and thus able to capture videos of the traffic signals ahead. This is just as users would mount their smartphone when using a navigation or other travel-related application.

5.1.1 Detection Algorithm

SignalGuru's traffic signal detection module must be lightweight and fast so that the color of traffic signals can be sensed as frequently as possible, and the time of transitions is detected as precisely as possible. The time accuracy of color transition detections directly affects the time accuracy of predictions, as Section 5.4 explains. Our SignalGuru detection module is able to process a fresh frame every two seconds.

Figure 2 shows our image processing algorithm used to process video frames for the purpose of detecting traffic signals. The algorithm is based on the three most characteristic features of a traffic signal, which are the bright red/yellow/green color of its bulbs, the shape of its bulbs (*e.g.,* circle, arrow) and its surrounding black box (traffic signal housing).

The first step of the detection algorithm is the color filtering process, as the most distinctive feature of traffic signals is the bright color of their bulbs. The color filter inspects the color of all pixels of an image (video frame) and zeroes out the pixels that could not belong to a red, yellow or green traffic signal bulb. Thus the color-filtered image contains only objects that have the correct color to be a traffic signal bulb. The color filter was designed empirically by analyzing the color range of red, yellow, green bulb pixels from a set of 400 traffic signal pictures[3]. This filter is fairly computationally-lightweight when performed in the device's native colorspace (*i.e.,* RGB). It also manages to zero out most of an image, reducing computing needs in subsequent stages. For all these reasons, the color filtering stage comes first.

After color filtering, only objects that have the correct color are maintained in the image. The next stages examine which of them qualify to be a traffic signal based on their shape (*e.g.,* circle, arrow). This is achieved by first applying a Laplace edge detection filter that highlights the boundaries of the color filtered objects and then a Hough transform. The Hough transform uses a voting mechanism (accumulator) to decide which objects constitute the best traffic signal bulb candidates based on their shape.

Once the Hough transform voting is completed, the accumulator determines which object has the most votes and is thus the best candidate to be a traffic signal. The accumulator contains information about the location of the best candidate in the image as well as its size (*e.g.,* radius).

Then, the pixels of the candidate area are inspected to decide on the color of the bulb and count exactly what percentage of the pixels falls into the correct color range. This percentage is termed the *Bulb Color Confidence (BCC)*. BCC helps to avoid confusing, for example, road signs with a circular red perimeter but a different color in the center (*e.g.,* right turn prohibited sign) as a red signal.

According to the color and size of the bulb, a specific area around the bulb is checked for the existence of a horizontal or vertical black box, the traffic signal housing. For example if the bulb is red, the area below or on the left is searched for a vertical or horizontal traffic signal black box, respectively. A *Black Box Confidence (BBC)* metric is also reported based on how many pixels of the searched area are dark enough to qualify as traffic signal black box pixels.

The product $BCC \times BBC$ constitutes the *detection confidence* for a specific object in the video frame. If the detection confidence is higher than a threshold value, then the detection is considered valid and the a traffic signal with the detected color is reported. If not, the next best candidate from the Hough transform accumulator is examined. We found that a detection confidence threshold of 0.6 yielded the lowest detection false positive and false negative rates for our database (400 pictures). We also found that there is little additional value in inspecting more than the 10 best candidates of the Hough voting mechanism. As a result, the looping criterion in Figure 2, N, is set to 10 for our work.

2. We configure the iPhone's GPS to return location stamps of the maximum possible accuracy and frequency.

3. We used a different color filter for Cambridge and Singapore as the two cities use traffic signals implemented with different technologies.

5.1.2 IMU-based Detection Window

For visibility and other practical reasons, traffic signals are placed high above the ground. As a result, traffic signals often appear only in the upper part of a captured video frame. As shown in Figure 4, the lower half of the image captures the road and other low-lying objects, whereas the traffic signals are located in the upper half. The part of the image where the traffic signal can be located depends not only on the orientation of the windshield-mounted smartphone, but also on the distance from the traffic signal; the closer the phone to the traffic signal, the higher the signal appears in the image.

SignalGuru leverages information from the smartphones' inertial sensors to narrow its *detection window*, *i.e.,* the part of the image where traffic signals are expected to appear. More specifically, SignalGuru uses information from the accelerometer and gyro-based Inertial Measurement Unit (IMU) of the smartphone to infer its orientation (roll angle) and information from its GPS device to calculate distance from the signal.

With this information, the size of the detection window can be easily calculated analytically. The calculation is left out in the interest of space. The IMU-based detection window is shown with a red bounding box in Figure 4.

The IMU-based detection window scheme enables SignalGuru to ignore a large portion of a captured frame that can have nothing but noise, providing twofold benefits. First, the image processing time is almost halved. Second, the traffic signal detection is significantly improved. The benefits of this scheme are evaluated in Section 7.2.

5.1.3 Variable Ambient Light Conditions

Ambient light conditions significantly affect the quality of captured still images and video frames. The amount of ambient light depends on both the time of the day and the prevailing weather conditions (sunny vs. cloudy). Smartphone cameras automatically and continuously adjust their camera exposure setting to better capture the target scene. Nevertheless, we found that traffic signals are often not captured well with their bulbs appearing either too dark (underexposed) or completely white (overexposed). As a result, the detection module would perform very poorly in some cases.

Traffic signals, however, have a fixed[4] luminous intensity. We leverage this by adjusting and locking the camera exposure time to the fixed intensity of traffic signals. This eliminates the sensitivity of traffic signal detection to time of day or weather. The camera exposure time is automatically adjusted by pressing the "Adjust Exposure" button and pointing the camera to a traffic signal. Then by pressing the "Lock Exposure" button the setting is recorded and locked, obviating the need for further adjustments.

5.2 Transition filtering module

The raw detection of traffic signals and their color transitions (R→G) given by the detection module is fairly noisy. In our Singapore deployment, in 65% of the cases that a vehicle is waiting at a red traffic signal, it reports a false positive transition i.e., a transition that did not actually occur. Typically, the image detection module was detecting the actual red light and then happened to misdetect some arbitrary object for a green light. Note that vehicles were waiting at the intersection for 48s, on average, capturing and processing perhaps dozens of video frames. If not handled properly, a single false green light detection would be enough to erroneously generate a transition report. Similarly, if a vehicle happens to misdetect an arbitrary object for a red light in between detections of the actual green light, a false transition would be reported.

While ideally we would like to be able to detect and report all R→G transitions witnessed (no false negatives), it is even more critical to avoid false positives (reports of transitions that never happened), because false positives pollute the prediction scheme. Therefore, we filter R→G transitions using a two-stage filter: a low-pass filter followed by a colocation filter.

5.2.1 Low Pass Filter (LPF)

According to our findings from our Singapore deployment, in 88% of the cases, false positive detections occur over a single frame and do not spread over multiple consecutive frames. As a result, most false transitions have one of the following three patterns with the false detection marked in bold:

1) R→ ...→R→**G**→R→...→R
2) G→ ...→G→**R**→G→...→G
3) NS→...→NS→**R**→**G**→NS→...→NS

The first (most common) pattern occurs when the vehicle is waiting at the red light it correctly detects, then at a specific instance it misdetects a passing object (*e.g.,* design on a bus crossing the intersection) for a green traffic light. The second pattern occurs when the vehicle misdetects an arbitrary object for a red light in between detections of the actual green light. Finally, the third pattern occurs when the view of the vehicle is obstructed and there is no traffic signal in sight. However, at some point, it misdetects an arbitrary object for a red light and right after that a different object for a green light. This pattern is the least common.

The LPF filters out such "spikes" or anomalies across multiple traffic signal transitions by adding some hysteresis. The LPF classifies only transitions that have the R→ R→G→G pattern as valid, *i.e.,* at least two red status reports followed by at least two green status reports. As our results in Section 7.3 show, the LPF filters the vast majority of false positive transitions at the cost of creating only a small number of false negatives (actual transitions removed by the filter).

5.2.2 Colocation Filter

A distinctive feature of traffic signals, as opposed to other objects with similar colors and shape, is that the red and the green bulb are contained in the same black box; that is, they are *colocated*. SignalGuru's filtering module leverages this by checking whether detected red and green bulbs are colocated before accepting a transition as valid. More specifically the colocation filter checks whether the green bulb that was just detected is close to the red bulb detected in the previous frame. Note that the accumulator of the Hough transform pinpoints the location of the traffic signal candidates.

4. LED traffic signals have fixed luminous intensity. Older incandescent traffic signals do not, but are quickly becoming obsolete. Both Cambridge and Singapore use LED traffic signals.

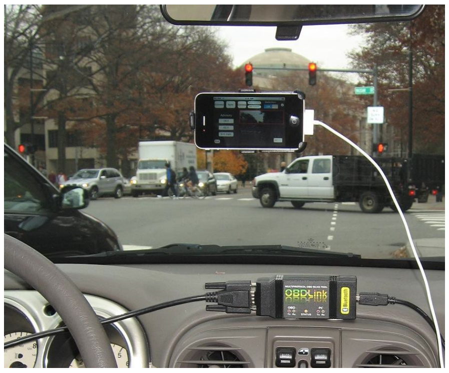


Fig. 3: **SignalGuru-enabled iPhone mounted on the windshield. The OBD-LINK device used to measure fuel consumption is also shown.**

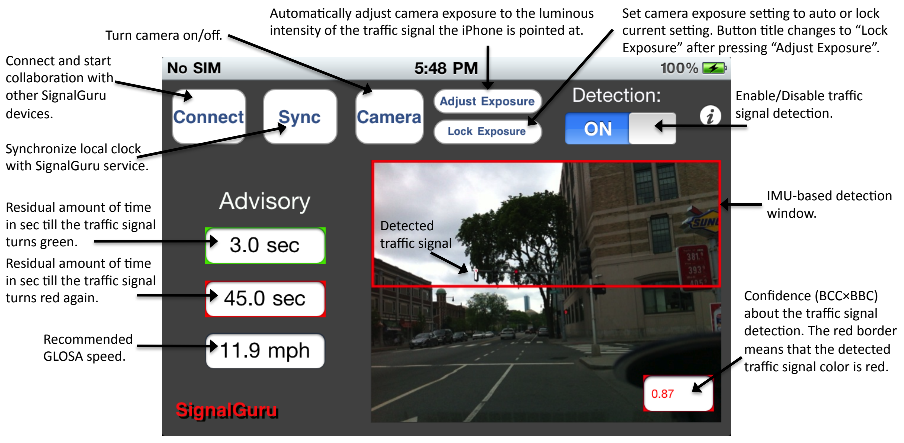

Turn camera on/off.
Automatically adjust camera exposure to the luminous intensity of the traffic signal the iPhone is pointed at.
Set camera exposure setting to auto or lock current setting. Button title changes to "Lock Exposure" after pressing "Adjust Exposure".
Connect and start collaboration with other SignalGuru devices.
Synchronize local clock with SignalGuru service.
Residual amount of time in sec till the traffic signal turns green.
Residual amount of time in sec till the traffic signal turns red again.
Recommended GLOSA speed.
Enable/Disable traffic signal detection.
IMU-based detection window.
Confidence (BCC×BBC) about the traffic signal detection. The red border means that the detected traffic signal color is red.
No SIM
5:48 PM
100%
Connect
Sync
Camera
Adjust Exposure
Lock Exposure
Detection:
ON
Advisory
3.0 sec
45.0 sec
11.9 mph
Detected traffic signal
0.87
SignalGuru

Fig. 4: **SignalGuru service screenshot. GLOSA advisory has also been included in the same iPhone application. Audio advisory can complement the visual advisory to alleviate driver distraction.**

Given that SignalGuru can capture and process video frames every 2s, the average delay between the light turning green and SignalGuru capturing this event in its next video frame is 1s. In 1s or even 2s that is the maximum possible green light capture delay, a vehicle will not have accelerated and moved significantly. Hence, there is no need to compensate for vehicle movement.

However, as exemplified in Figure 3, many intersections have two or more traffic signals for the same phase or direction of traffic. Therefore, the red and green bulbs may be detected on different traffic signals across the two frames. To tackle this, before the colocation filter rejects a transition as invalid, it invokes the detection module to check whether there exists, in the current frame, a green bulb that is collocated with the red bulb of the previous frame. In this case, the detection window covers only a very small area around the red traffic signal of the previous frame, and thus incurs negligible computational overhead.

As shown in Section 7.3, the colocation filter effectively filters out false positive transitions at the cost of a small increase in false negatives. Together, the LPF and the colocation filter form a very robust two-stage filter.

5.3 Collaboration module

SignalGuru depends on the grassroots collaboration among the participating nodes (smartphones). A node is limited by its field of vision, and does not have all the information it needs in order to predict the schedule of the traffic signals ahead. Typically, a node needs information about a traffic signal well before the signal comes into the node's camera field of view.

For the prediction of traffic-adaptive traffic signals, collaboration is even more critical. As we explain in Section 5.4.2, in order to be able to predict traffic-adaptive traffic signals, information from all phases (intersecting roads) of an intersection is needed. Furthermore, Section 7.4.3 shows how more collaborating nodes and more traffic signal history can improve the prediction accuracy for the challenging traffic-adaptive traffic signals of Singapore.

The collaboration module allows participating SignalGuru nodes to opportunistically exchange their traffic signal information (timestamped R→G transitions) by periodically (every two seconds) broadcasting UDP packets in 802.11 ad-hoc mode. A SignalGuru node broadcasts not only the data it has sensed on its own, but also the data it has opportunistically collected so far. Only data about the traffic signal transitions of the last five cycles is exchanged. We found that using a longer history of data does not significantly improve the traffic signal prediction accuracy.

In order to be able to predict the schedule of the traffic signals ahead, nodes need either the database of the traffic signal settings (for pre-timed traffic signals) or the Support Vector Regression (SVR) prediction models (for traffic-adaptive signals). This information is passed to a node along with the sensed transition data before the node approaches the corresponding traffic signals. However, it is likely that this node will have also crossed the traffic signal ahead in the recent past (e.g., yesterday due to daily commute). In this case, the sizeable (62 KB) SVR prediction models do not need to be resent as they are relatively static (Section 7.4.3). The settings for a pre-timed traffic signal can be encoded in just a couple bytes and thus resending them incurs negligible overhead.

The amount of traffic signal information[5] that SignalGuru nodes gather and exchange can be constrained by tiling a geographic area into regions and having SignalGuru nodes maintain and exchange data that belongs only to their current region. Methods for tiling a region into sub-regions for the purpose of ensuring data availability and load balancing are beyond the scope of this paper. Furthermore, the aggregate regional resources for the operation and maintenance of the SignalGuru service can be kept at bay by running SignalGuru atop resource-aware middleware like RegReS [16].

5.4 Prediction module

Two main categories of traffic signals exist: pre-timed and traffic-adaptive traffic signals. Since their operation is very different, SignalGuru uses different prediction schemes for each category.

5. Sensed traffic signal transitions, database of traffic signal settings and SVR prediction models for pre-timed and traffic-adaptive traffic signals, respectively.

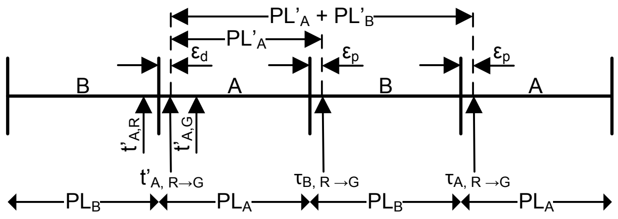


Fig. 5: **Timeline of traffic signals operation and SignalGuru's detections and predictions for a simple intersection with two phases (A and B). The letters on the timeline denote for which of the two phases the light is green. The timestamps of actual, detected and predicted phase transitions are also marked with t, t' and τ, respectively. PL$_A$ is the actual length of phase A and PL'$_A$ its predicted value. ε_d and ε_p are the color transition detection and prediction errors, respectively.**

5.4.1 Pre-timed Traffic Signals

SignalGuru's prediction module maintains a database of the traffic signal settings. As described in Section 4, pre-timed traffic signals have fixed pre-programmed settings for their different modes (a.m./p.m. peak, off-peak, Saturday peak). Traffic signal settings can be acquired from city transportation authorities. In case they are not available, the settings (phase lengths) can be measured as described in Section 5.4.2. This means that SignalGuru knows how long each phase lasts. The challenge remains to accurately synchronize SignalGuru's clock with the time of phase transition of a traffic signal. Once this is achieved, SignalGuru can very easily predict when the traffic signal will switch again to green, yellow or red.

Clock synchronization is achieved by capturing a color transition, *e.g.,* R→G. Figure 5 shows a timeline of events. If the timestamps of the last red and first green color detections for phase A are $t'_{A,R}$ and $t'_{A,G}$, respectively, then the detected transition time is $t'_{A,R\to G} = (t'_{A,R} + t'_{A,G})/2$.

The time the traffic signal will switch to red for phase A (and green for phase B) can be predicted by adding the predicted[6] length of phase A (PL'_A) to $t'_{A,R\to G}$ as $\tau_{B,R\to G} = t'_{A,R\to G} + PL'_A$. Since this intersection has only two phases, phase A will follow after phase B; as phases are scheduled in a predictable, round-robin way. By adding (PL'_B) to $\tau_{B,R\to G}$, we get the next R→G transition for phase A, and so on.

Clock synchronization needs to be reestablished only after the traffic signal changes mode of operation or recovers from an operational failure. As a result, the necessary participation requirements for the detection of pre-timed traffic signals is very low. Only a few vehicles are required, as traffic signal transitions need to be detected only once every a couple hours or potentially at even lower frequencies. Unless a cloud server is used, more vehicles may be necessary, however, for the maintenance of SignalGuru's data around the area of the corresponding traffic signals.

5.4.2 Traffic-adaptive Traffic Signals

The Singapore GLIDE (SCATS) system is one of the world's most sophisticated and dynamic traffic-adaptive traffic signal control systems. As described in Section 4, SCATS measures the saturation of intersection phases and adjusts their phase lengths at every cycle. Phase lengths change when SCATS changes the value of the cycle length or the fraction of the cycle length that gets allocated to them. SCATS may choose to change both settings at the same time. Phases are still scheduled in a deterministic round-robin manner.

SignalGuru predicts future transitions (*e.g.,* when the signal ahead will turn green) by detecting past transitions and predicting the length of the current or next phases. The key difference from the prediction of pre-timed traffic signals lies in the prediction of the *phase length*, as opposed to looking it up from a database.

SignalGuru predicts the length of a phase by measuring and collaboratively collecting the prior traffic signal transition history, and feeding it to a Support Vector Regression (SVR) [4] prediction model. In Section 7.4.3, we evaluate the prediction performance of different Prediction Schemes (PS) by training the SVR with different sets of features:

- PS1: The past lengths of the specific phase. For example, the next length of phase A is predicted based on history information such as the lengths of the five previous phases of A. We found that further increasing the length of the history does not yield any benefits. Similarly, SCATS uses only loop detector measurements performed over the last five cycles to determine the next traffic signal settings.
- PS2: Like PS1, but the length of the preceding phases of the same cycle is also provided. This means that when trying to predict the length of phase C, the lengths of preceding phases A and B are also fed to the SVR model. As our results show, this information significantly improves the performance of the prediction module. The reason is that changes to a given cycle's phase lengths are correlated; when SCATS changes the cycle length setting, all phases change in the same way.
- PS3: Like PS2, except that information for the past 5 cycle lengths is also factored in.
- PS4: In addition to PS3's features, this scheme assumes the predictor has access to loop detector saturation information, which is not currently feasible. Saturation values over the past 5 cycles are fed to the SVR model. Note that traffic (vehicle speed) estimation is not a good proxy for the unavailable loop detector measurements. Average vehicle speed does not always correlate well with the saturation measured by SCATS's loop detectors; a specific phase, despite the fact that vehicles are moving fast, may be highly saturated (with dense flow).

In order for SignalGuru to be able to use any of the first three feasible prediction schemes, the lengths of the past phases need to be measured. While it is easy for SignalGuru to detect the $R \to G$ transition for the beginning of a phase, as explained in Section 4, it is very hard to detect the $G \to Y$ transition for the end of the phase. To remedy that, collaboration across nodes waiting at the different traffic signals of the same intersection is leveraged; the $G \to Y$ transition of a given phase is inferred by the $R \to G$ transition of the successor phase that was detected by nodes waiting at the traffic signal of

6. For pre-timed traffic signals, the predicted phase length is the value looked up in the traffic signal settings database.

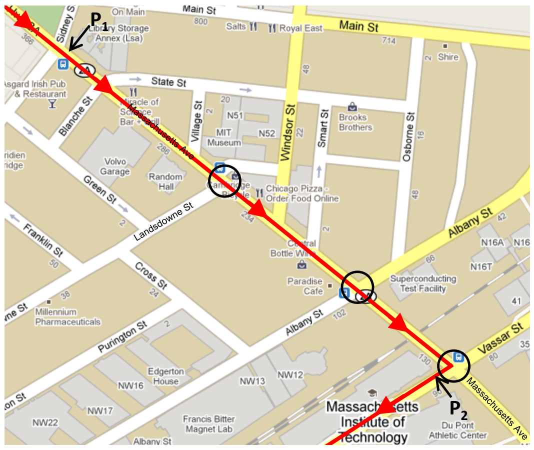


Fig. 6: **Route of vehicles in Cambridge deployment. The targeted intersections are marked with circles. P_1 and P_2 are the start and end points, respectively, for our GLOSA experiment trip.**

the successor phase. For example, the fact that the light turned green for phase B at time t means that it turned yellow for phase A at time *t* minus the *clearance interval*. The clearance interval is a fixed setting and is the amount of time a phase is yellow plus the amount of time all phases are red before the next one turns green. As its name denotes, it gives the previous phase enough time to clear before the next conflicting phase starts.

A week of history data is enough to train the SVR model. Furthermore, as our results show, the SVR model does not need to get continuously re-trained. Re-training the model every 4 to 8 months is frequent enough in order to keep the prediction errors small.

On the other hand, real-time (on a minute-by-minute scale) traffic signal transition data are necessary in order to effectively drive the prediction models. As we show in Section 7.4.3, the less complete and the older the available historic information is, the lower the prediction accuracy becomes. Because of the highly dynamic nature of traffic-adaptive traffic signals, significantly higher participation (about two orders of magnitude) is required for them as compared to pre-timed traffic signals.

6 METHODOLOGY

6.1 Cambridge Deployment

As mapped in Figure 6, our November 2010 deployment in Cambridge targeted three consecutive intersections on Massachusetts Avenue. We used 5 vehicles with iPhones mounted on their windshields and asked the drivers to follow the route shown for ∼3 hours. Note that the opportunity for node encounters (within ad-hoc wireless range) was small, as all the vehicles followed the same route so they are rarely in range of each other. To rectify this, an extra iPhone device was held by a pedestrian participant located at the intersection of Massachusetts Avenue and Landsdowne Street. This SignalGuru device served as an ad-hoc data relay node facilitating data exchange between the windshield-mounted iPhone nodes. Only the collaboration module was active on the relay node. The experiment took place between 1:20pm - 4:30pm. At 3:00pm, the traffic signals changed operation mode from off-peak to afternoon peak.

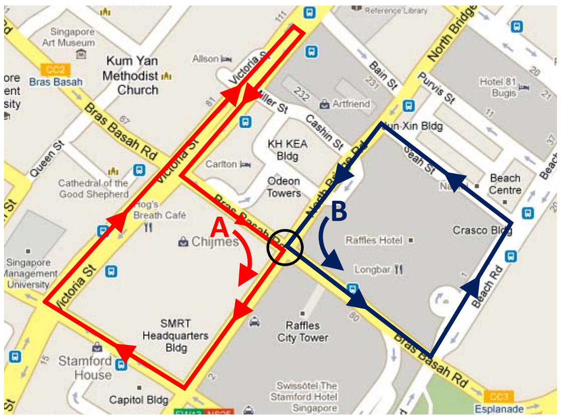


Fig. 7: **The two distinct routes of taxis in Singapore deployment in the Bugis downtown area. Routes A and B correspond to phases A and B of the targeted intersection, respectively. The targeted intersection is marked with a circle.**

6.2 Singapore Deployment (Bugis Downtown Area)

Our other deployment was in Singapore in August 2010. Unlike Cambridge, the Singapore deployment tests SignalGuru on traffic-adaptive traffic signals. To measure phase lengths and predict the schedule of traffic-adaptive traffic signals, SignalGuru needs to monitor all phases of an intersection, *i.e.,* orthogonal directions of a traffic intersection. Hence, in this deployment we had two sets of vehicles following the two distinct routes shown in Figure 7. In this way, both phases of the intersection (Bras Basah and North Bridge Road in Singapore's downtown) were sensed. Phase A corresponds to vehicles moving along Bras Basah Road and phase B to vehicles moving along North Bridge Road.

We used eight iPhone devices in total and mounted them on the windshields of taxis. Five devices were moving on the longer route of phase A and the other three on the shorter route of phase B. Similarly to our deployment in Cambridge, an extra iPhone device was used as a relay node. In this case, the relay node was also recording the ground truth[7], *i.e.,* when the traffic signals status transitioned. Ground truth information was only used for offline evaluation of SignalGuru's accuracy thereafter. It was not shared with other participating nodes. The experiment took place from 11:02am - 11:31am (∼30min).

7 SIGNALGURU EVALUATION

Here, we evaluate the performance of each of SignalGuru's modules before evaluating its overall performance in two deployments in Cambridge and Singapore. We also performed a large-scale analysis for SignalGuru's prediction accuracy based on the data we collected from Singapore's Land Transport Authority.

7. In our Cambridge deployment, since the schedule of the signals is fixed, it can be easily inferred from the images logged by the windshield-mount iPhones. Hence, there was no need to record the ground truth with an extra iPhone device.

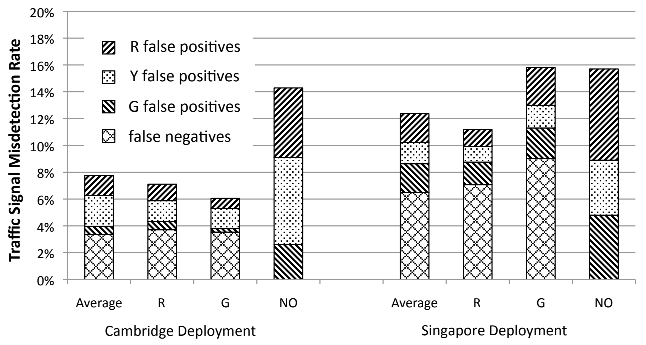


Fig. 8: **Traffic signal detection module evaluation. R/Y/G/NO stands for video frames where the traffic signal is actually Red/Yellow/Green or non-existent. A false negative is when the module fails to detect the existing traffic signal. A false positive is when the module confuses an arbitrary object for a traffic signal of a specific R/Y/G status. We omitted "Y" results as there were very few such frames and hence the detection results are not statistically important.**

7.1 Traffic Signal Detection

We evaluate the performance of SignalGuru's detection module for our two deployments. In Figure 8, we show both the percentage of false negatives (traffic signals that did not get detected) and the percentage of false positives (arbitrary objects confused for traffic signals of a specific color). Results are averaged over 5959 frames and 1352 frames for the Cambridge and Singapore deployments, respectively. The average misdetection rate that includes both false negatives and false positives was 7.8% for Cambridge and 12.4% for Singapore deployment. In other words, SignalGuru's detection module correctly detected the existence (or the lack) of a traffic signal in 92.2% and 87.6% of the cases in Cambridge and Singapore, respectively. Note that most (>70%) video frames are captured while vehicles are waiting at the red light. Hence, the average (mis)detection rate is strongly biased by the results for "R", *i.e.,* frames with a red traffic signal.

As Figure 8 shows, the detection module is particularly more likely to report a false positive when there is no traffic signal in sight. When a traffic signal is captured in the video frame, the actual traffic signal will normally get the most votes in the Hough transform's accumulator and a valid detection will be recorded. If there is no traffic signal in sight, the detection module will try to find the best possible candidate object that most resembles a traffic signal in terms of its color, shape and enclosing black box, which can trigger more false positives.

Furthermore, the ratio of false positives of different colors differs significantly across the two deployments. For example in Cambridge, yellow light false positives are more common than in Singapore, where there are more green light false positives. This is because of the prevailing ambient light conditions and the object composition of the environment at the targeted intersections. In Singapore, there were many more trees and also a black electronic message board with green letters, whereas in Cambridge, the sun was setting, giving a strong yellow glare to several objects (*e.g.,* road signs, vehicles, buildings, *etc.*).

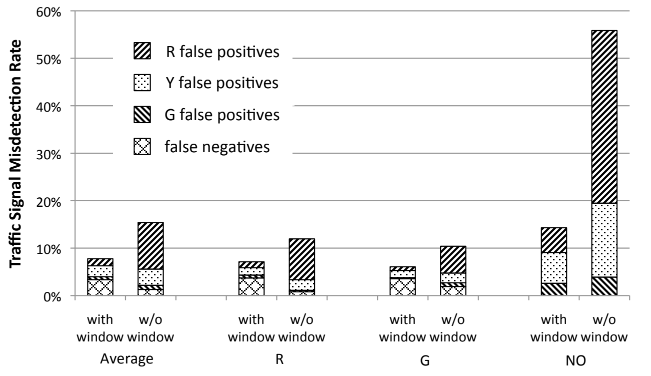


Fig. 9: **IMU-based detection window scheme evaluation for Cambridge deployment. The IMU-based detection window scheme almost halves the rate of misdetections.**

Another interesting observation is that the number of false negatives (missed traffic signal detections) is almost double in the Singapore deployment, as compared to the Cambridge deployment. The reason lies in the traffic signal bulbs used in each city. Singapore's LED bulbs are exposed, whereas Cambridge's are covered by a refraction lens. The LED traffic signal bulbs consist of an array of smaller LEDs that is refreshed in columns at a relatively low frequency. The refresh frequency is high enough to be invisible to the human eye but low enough to be detectable by a camera when there is no refraction lens covering the bulb. In Singapore, the camera would thus sometimes capture the bulbs with dark stripes (columns) of unrefreshed LEDs, reducing the probability of a successful traffic signal detection.

7.2 IMU-based Detection Window

In this section, we evaluate the benefits that the IMU-based detection window offers. The iPhones were oriented horizontally, as shown in Figure 3. The lower line of the detection window will thus be horizontal and across the center of the image when the vehicle is at a distance of ∼50m from the intersection. Using online/offline traffic signal detection, we collected results for when the IMU-based detection window was activated/deactivated. The offline detection was based on the same video frames that were logged and processed by the iPhone devices online.

Figure 9 shows that the IMU-based detection window almost halves the average misdetection rate, reducing it from 15.4% to 7.8%. Above all, the IMU-based detection window significantly reduces the number of red false positives; when the detection window scheme is not used and the whole video frame is processed, the detection module often confuses vehicles' rear stop lights for red traffic signal bulbs.

On the other hand, the IMU-based detection window scheme increases the number of false negatives when the traffic signal is red. When a vehicle is decelerating abruptly to stop at the red light, the IMU miscalculates the device's orientation. As a result, the detection window is also miscalculated, becoming so small that the traffic signal is excluded. Nevertheless, the

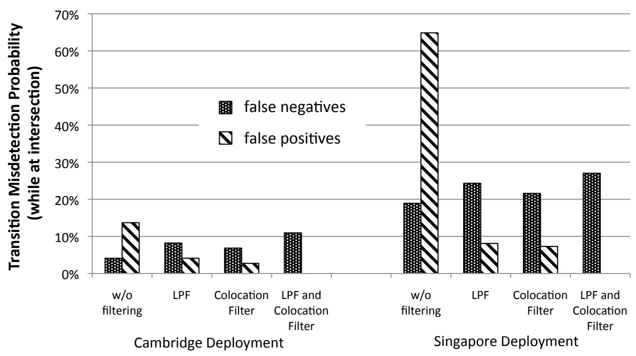


Fig. 10: **Transition filtering module evaluation. The transition filtering module removes false positive reports without significantly increasing the number of false negatives. The iPhone devices witnessed 219 and 37 traffic signal transitions in our Cambridge and Singapore deployments, respectively.**

effects of abrupt decelerations are only transient and a car is soon able to detect the traffic signal ahead.

Finally, since only a fraction of the video frame is processed, the IMU-based detection window scheme reduces the average processing time by 41% (from 1.73s to 1.02s).

7.3 Transition Filtering

The performance of the transition filtering module is evaluated in terms of the number of false positives (invalid transitions) it manages to remove and the number of false negatives it creates (valid transitions erroneously removed).

As shown in Figure 10, the probability of (unfiltered) false positives in the Cambridge deployment is significantly smaller when compared to the Singapore deployment. This occurs for two reasons: First, the rate of false positive traffic signal detections is smaller in Cambridge. Second, the average waiting time at red traffic signals is only 19.7s for Cambridge vs. 47.6s for Singapore. As a result, the probability of a false positive transition detection during that waiting time is significantly lower.

While the LPF and colocation filters each significantly reduce the number of false positives, it is when both filters are applied in series that all false positives are removed in both deployments, with only a small increase in the number of false negatives. More specifically, the probability of false negatives increased by 6.8% for Cambridge and 8.1% for Singapore. Thus, the transition filtering module effectively compensates for our lightweight but noisy traffic signal detection module.

7.4 Schedule Prediction

7.4.1 Cambridge deployment

We evaluate the overall accuracy of SignalGuru's traffic signal schedule predictions for Cambridge's pre-timed traffic signals. As evaluation metric, we use the prediction mean absolute error; the absolute error between the predicted and the actual traffic signal phase transition time, averaged across the 211 predictions performed by the participating iPhone devices.

As shown in Figure 11, SignalGuru can predict the schedule of pre-timed traffic signals with an average error of only 0.66s.

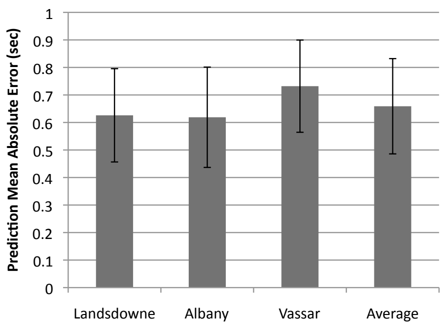


Fig. 11: **Mean absolute error of SignalGuru's traffic signal schedule predictions for the three targeted intersections in Cambridge. The error bars show the standard deviation of the mean absolute error. The ground truth on the status of traffic signals was inferred by the images logged by the windshield-mounted iPhones with sub-300ms accuracy.**

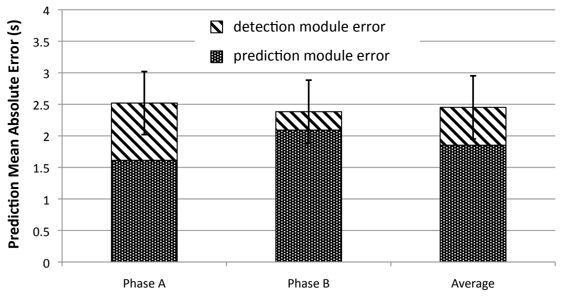


Fig. 12: **Traffic signal schedule prediction evaluation for Singapore deployment. The ground truth was recorded every two seconds and the actual (ground truth) transition time for a phase,** *e.g.,* **A, was calculated as $t'_{A,R\to G} = (t'_{A,R} + t'_{A,G})/2$. The measurement error was thus 1s (shown with error bars).**

Since SignalGuru uses a database for the settings of pre-timed traffic signals, the prediction error is solely caused by the error with which SignalGuru detects color (phase) transitions. When SignalGuru captures and processes video frames every T=2s, the transitions are theoretically detected with an error that has a maximum value of ε_{max}=T/2=1s and an expected value of $[\varepsilon]$=T/4=0.5s. This is very close to the measured prediction error value of 0.66s. With this very small prediction error, SignalGuru can effectively support the accuracy requirements of all applications described in Section 3.

7.4.2 Singapore deployment

We evaluate the accuracy of SignalGuru's traffic signal schedule predictions for Singapore's traffic-adaptive traffic signals, using the prediction mean absolute error as the evaluation metric. The prediction module was configured to use the prediction scheme PS3, and was trained offline using a week's worth of data (June 1-7 2010) that we obtained from Singapore's Land Transport Authority (LTA).

As our results in Figure 12 show, SignalGuru can predict the time of the next color transition with an average error of 2.45s. The next color transition prediction error is broken down into an average absolute error of 0.60s in detecting the current phase's start time (detection module error) and an average absolute error of 1.85s in predicting the length of the

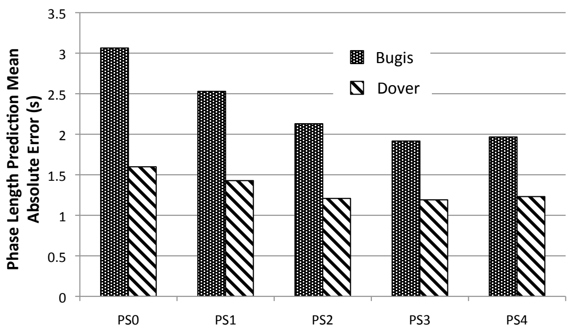


Fig. 13: **Evaluation of the different prediction schemes for Bugis and Dover traffic signals.**

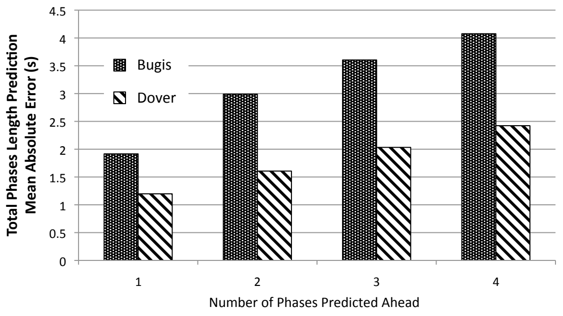


Fig. 14: **Evaluation of SignalGuru's prediction scheme PS3 when predicting multiple phases ahead for Bugis and Dover traffic signals.**

current phase (prediction module error). The prediction error is due to both the inaccurate phase duration measurements that are fed into the SVR model and the prediction error of the SVR model, with the latter being the main contributor. The phase duration measurement error has a triangular probability density function[8] and the expected value for the phase duration measurement absolute error is only [$\varepsilon_{duration}$]=T/3=0.66s. Results are averaged over 26 predictions. The schedule prediction accuracy for the two phases is comparable.

Without collaboration and high enough participation, SignalGuru would not be able to measure the past length of the phases for the purpose of feeding them into the SVR-based prediction scheme and predicting their future lengths. SignalGuru would have to predict the future phase length for a traffic-adaptive traffic signal using the same scheme that it uses for pre-timed signals i.e., by using a typical fixed value for it. Such a value could be, for example, the average length of the phase during the same hour of the previous day. In this case, the prediction error. for our Singapore deployment would have been 11.03s instead of 2.45s (3.5X higher). In the next section, we evaluate the performance of SignalGuru for different degrees of participation.

7.4.3 *Singapore Large-scale Prediction Analysis*

In order to perform a large-scale evaluation of the performance of SignalGuru's prediction module across different traffic signals and intersections with different traffic patterns, we collected traffic signal operation logs from Singapore's Land Transport Authority. More specifically, we collected logs for 32 traffic signals (phases) in the Dover (suburban) area and for 20 traffic signals (phases) in the Bugis (downtown) area. The logs spanned over the two weeks of June 1-14, 2010 and contained more than 200,000 phase lengths for both Bugis and Dover traffic signals. We used the logs of the first week to train the different SVR-based prediction schemes, and the logs of the second week to test their performance. The training and testing sets were therefore not overlapping.

Prediction Schemes Evaluation. In Figure 13, we evaluate the performance of the different phase length prediction schemes for the traffic signals of Dover and Bugis. We also include the performance of a baseline scheme "PS0" that uses the last measurement of a phase's length as the prediction for its future length. PS3 outperforms PS1 and PS2 and reduces the phase length prediction mean absolute error by 37% (from 3.06s to 1.92s) for Bugis and by 26% (from 1.60s to 1.19s) for Dover when compared to PS0.

As shown in Figure 13, the prediction mean absolute error for Dover traffic signals is half when compared to the error for Bugis traffic signals. However, note that the average phase length for Bugis is 47s whereas for Dover it is only 28s. As a result, the relative errors are more comparable: 4.1% for Bugis and 4.3% for Dover.

Surprisingly, we found that the theoretical prediction scheme PS4, which assumes knowledge of loop detector information, does not outperform PS3. We believe that this is because the effects of loop detector measurements are already captured by SCATS in the history of the phase and cycle length settings that it chooses. SignalGuru measures them and uses them as prediction features for PS3.

Increasing available lead-up time. In order to increase the available lead-up time beyond the length of a single phase[9], SignalGuru needs to predict multiple phases ahead. For traffic-adaptive traffic signals, the prediction error increases as SignalGuru tries to predict multiple phase lengths ahead. For pre-timed traffic signals, for which the phase lengths are fixed and known, the prediction error only depends on the ability of SignalGuru to synchronize accurately with the traffic signal and thus lead-up time is arbitrarily long so long as it is within the same traffic mode.

Figure 14 shows the error of the prediction module, when it predicts the lengths of multiple phases ahead. The prediction error increases sublinearly as the number of predicted phases increases. However, even when predicting four phases ahead, the total prediction error for all phase lengths is only 4.1s (8.7%) and 2.4s (5.2%) for Bugis and Dover traffic signals, respectively. Given that wireless 802.11g can broadcast a kilobyte of data over several hops in <1s, the average available lead-up times for Bugis and Dover are 187s and 114s, respectively. The percentage of available data (% transitions detected) in our Singapore deployment was 81%.

As this analysis shows, SignalGuru can predict accurately the schedule of traffic-adaptive traffic signals regardless of their location, *e.g.,* suburban or downtown. Furthermore, their schedule can be predicted multiple phases in advance with

8. Assuming errors for the detection of phase start and stop times are independent and uniformly distributed in [0, T/2].

9. Predicting a single phase in advance suffices for all proposed applications except TSAN.

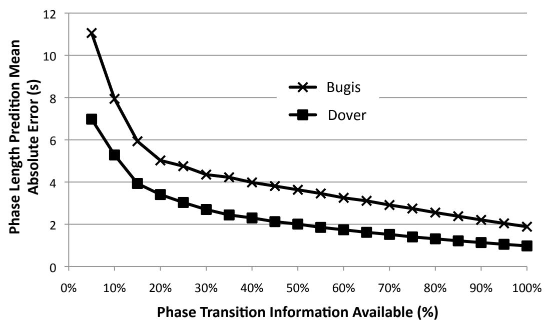


Fig. 15: **Phase length prediction accuracy for Bugis and Dover traffic signals as the percentage of the available traffic signal transition data varies.**

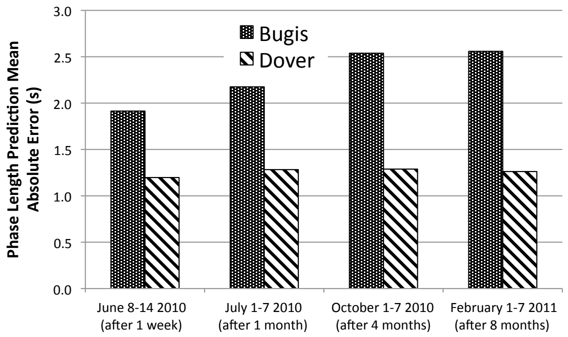


Fig. 16: **Prediction model performance over time. The prediction performance of the SVR model that was trained with the data of June 1-7 2010 is evaluated for the weeks of June 8-14 2010, July 1-7 2010, October 1-7 2010 and February 1-7 2011.**

small errors, enabling all the novel applications mentioned in Section 3 for traffic-adaptive traffic signals.

Collaboration Benefits. Figure 15 shows how the accuracy of phase length predictions depends on the data availability. Namely, accuracy depends on the percentage of traffic signal transitions that are detected and made available (through collaborative sensing and sharing). Where the phase length cannot be determined (because no SignalGuru node detected its start or end), we used the previously predicted phase length. The more transition data is available (higher degree of collaboration), the better SignalGuru's prediction accuracy. When data availability drops below 25% for Bugis and 28% for Dover, relative prediction errors degrade to >10%. As a result, SignalGuru can no longer meet the requirements of the described applications. Collaboration is thus critical to ensure high-quality predictions.

SVR re-train frequency. We evaluate how well the SVR model that was trained using the data of June 1-7, 2010 can predict the schedule of the traffic signals after one week (June 8-14, 2010), one month (July 1-7, 2010), four months (October 1-7, 2010) and eight months (February 1-7, 2011). As shown in Figure 16, the SVR model can make accurate predictions even after 8 months. More specifically, the error for Dover traffic signals does not significantly increase over time. In contrast, for Bugis traffic signals, the prediction error increases by 33% (from 1.9s to 2.6s) after 8 months. LTA engineers manually perform changes to the traffic signal settings (*e.g.*, phase programs) over time in an attempt to better optimize the traffic signals operation in the busy Singapore downtown area. As a result, SingalGuru's prediction ability degrades over time for Bugis, and the SVR model needs to get retrained every couple months in order to keep prediction errors low.

TABLE 1: **Computation resources required by SignalGuru's different modules. The computation resources for the traffic signal detection module is further broken down in Table 2.**

	% Application CPU	% System CPU	CPU Time (sec)
Traffic Signal Detection	66.57	51.12	1.02
Logging	21.94	16.85	0.34
Image Format Conversions	2.85	2.19	0.04
Communication	1.35	1.04	0.02
Schedule Prediction	0.52	0.40	0.01
Misc. (display, etc.)	6.77	5.20	0.10
Total	100.00	76.79	1.53

TABLE 2: **Computation resources for the different steps of SignalGuru's traffic signal detection algorithm.**

	% Application CPU	% System CPU	CPU Time (sec)
Color Filtering	7.71	5.92	0.12
Laplace Edge Detection	3.59	2.76	0.06
Hough Transform	30.55	23.46	0.47
Find max in Accumulator	21.64	16.62	0.33
Misc.	3.07	2.36	0.05
Total	66.57	51.12	1.02

7.5 SignalGuru Service Overhead

In this section, we present the computational (CPU, memory) and communication resources that SignalGuru consumes. We profiled SignalGuru across one hour using Xcode's CPU Sampler performance tool. During the profiling, the iPhone 3GS device was facing the first intersection of Figure 6 at a distance of $\sim$30m from the traffic signal. SignalGuru was configured to process a new frame every 2 seconds using the IMU-based detection window scheme. The GPS and IMU modules were thus activated.

In Table 1, we show the computation time for the different components of SignalGuru. About 67% of the application CPU time is spent for the traffic signal detection. This corresponds to 51% of system CPU time. The logging of images takes up 22% of the application CPU time[10]. For pre-timed traffic signals, the traffic signal schedule prediction takes less than 10ms. For traffic-adaptive traffic signals, the prediction is based on the SVR models and takes 21ms.

Although SignalGuru is using only 77% of the system CPU time in this configuration, we could not further increase the video frame capture and traffic signal detection frequency. We found that when increasing this frequency by 20%, the GPS location updates would become very infrequent (<0.1Hz). This had detrimental effects as SignalGuru's traffic signal detection was not activated/deactivated promptly on approaching/leaving a signalized intersection.

The average memory footprint of SignalGuru was 120.0 MB. However, only 20.1 MB, on average, were kept in actual RAM. The remaining 99.9 MB were assigned by

10. Image logging was necessary in our experiments so that we can test whether the traffic signal detection was successful or not. In a real system, image logging would be disabled.

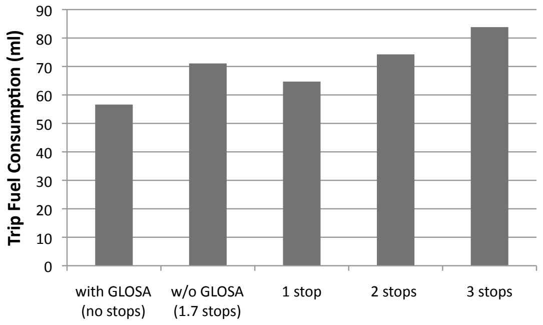


Fig. 17: **GLOSA fuel efficiency evaluation.**

the iOS to virtual memory. The iPhone 3GS devices have 256 MB of eDRAM. SignalGuru thus takes up only ~8% of the device's actual RAM resources.

The communication resources required, per smartphone, to collaboratively support SignalGuru are limited. Less than 1 KB/s and 22 KB/s are exchanged for pre-timed and traffic-adaptive traffic signals, respectively. The difference lies in the sizeable SVR models that are exchanged in the case of traffic-adaptive traffic signals.

7.6 GLOSA Fuel Efficiency

For evaluating GLOSA, we used a 2.4L Chrysler PT Cruiser '01 city vehicle. We measured its fuel efficiency by connecting to its Controller Area Network (CAN) with a Scan Tool OBD-LINK device (Figure 3). The fuel efficiency was calculated based on the Intake Manifold Absolute Pressure (IMAP) approach with the help of the OBDwiz software. The trip starts at P_1 and ends at P_2 on Figure 6, including the three intersections in our Cambridge deployment.

The driver completed 20 trips, following GLOSA's accurate advisory (< 1s mean absolute prediction error[11]) at odd-numbered trips, and driving normally (without GLOSA's advisory) at even numbered trips. When following GLOSA's advisory, the driver was able to avoid most stops (the driver sometimes had to brake because of pedestrians or cars in front). When not, the driver had to stop from zero to three times during each of the trips. As shown in Figure 17, GLOSA can offer significant savings reducing fuel consumption, on average, by 20.3% (from 71.1ml to 56.6ml). In other words, GLOSA improves the vehicle's mileage, on average, by 24.5% (from 16.1 mpg to 20.1 mpg).

8 Complex Intersections

In this section, we discuss practical issues regarding the operation of SignalGuru in complex intersections, as well as how SignalGuru can overcome them. In a complex intersection with many traffic signals, SignalGuru must be able to detect the correct traffic signal and also identify to which vehicular movement the detected traffic signal corresponds.

11. For small distances from traffic signals we found that it is more beneficial to provide the driver with the transition time instead of the recommended speed. First, for small distances (<50m) the GPS error is significant and second, the driver can better account for vehicles stopped at the intersection and time their acceleration appropriately.

8.1 Traffic Signal Detection

In a complex intersection with many traffic signals, SignalGuru will normally still detect the correct traffic signal, *i.e.,* the one that corresponds to the road segment that the vehicle is currently on. Normally, a vehicle that is approaching an intersection on a given road segment, will be able to view only the corresponding traffic signal at a zero-degree angle. The traffic signals of the other road segments may be still within the camera's field of view, but will be seen at some angle. At angles $> 90^\circ$ the traffic signal bulbs will not be visible. At smaller angles, the bulbs will be visible but will recorded on the video frame as ovals instead of circles. Furthermore, these ovals will be partially occluded by the traffic signal housing visors. While SignalGuru can still detect partially deformed and occluded traffic signals, its Hough transform voting algorithm will favor the most round and least occluded traffic signal *i.e.,* the one that corresponds to the road segment that the vehicle is currently on.

Moreover, information about the exact location of traffic signals at an intersection can be leveraged to further narrow down the size of the IMU-based detection window. In this way, both the accuracy and the speed of the traffic signal detection will get improved. The locations of the traffic signals can be detected and recorded by the SignalGuru devices themselves.

8.2 Traffic Signal Identification

In a complex intersection with more than one traffic signal, SignalGuru needs to identify which specific traffic signal it is detecting, *i.e.,* to which direction of movement the detected traffic signal corresponds. While GPS localization can be used to identify the intersection, it is often not accurate enough to distinguish between the different road segments of an intersection. The identification of the traffic signal being detected is necessary in order to appropriately merge the data detected across different vehicles.

The traffic signal is identified based on the GPS heading (direction of movement) of the vehicle that is detecting it, as well as the shape of the traffic signal (round, right/left turn arrow, *etc.*). The heading of the vehicle is used to identify the road segment on which the vehicle is located by matching its reported GPS heading ($d^\circ + \delta^\circ$, where δ° is the measurement error) to road segment that has the closest orientation (d°). The number and orientation of the intersecting road segments at any given intersection can either be acquired by mapping information [23] or learnt by clustering movement traces of SignalGuru-enabled vehicles. Then, for example, a vehicle can tell that the signal it is detecting is for vehicles that want to turn left and are approaching the intersection on the road segment that is attached to the intersection at d° degrees compared to geographic north.

9 Collaborative Services

Besides SignalGuru, windshield-mounted smartphones can support a rich set of novel collaborative services with their cameras. We briefly describe here four additional services:

Parking space availability service. Windshield-mounted smartphone cameras can process captured images to discover available parking spots and subsequently disseminate information about their location to drivers that are looking for a free spot. Free parking spots can be discovered by detecting features like the wheels of parked cars and the lines that segment parking spots.

Bus localization and arrival time service. Windshield-mounted smartphones could be used to support a grassroots bus localization and arrival time service. Windshield-mounted smartphone cameras can detect and identify buses by detecting their license plates and IDs. Bus IDs are often displayed with bright LED signs, making their detection and identification task significantly easier. On encountering and identifying a bus, smartphones can estimate and disseminate the bus arrival time based on the their location and the prevailing traffic conditions [29].

Taxi discovery service. In some cities (*e.g.,* Singapore), taxis have LED signs on their roof that show whether the taxi is free ("TAXI" shown in green) or busy ("BUSY" shown in red). Windshield-mounted smartphones could detect the color-coded text displays to discover free taxis and inform prospective passengers about where they can find one.

Cheap gas advisory service. Windshield-mounted smartphone cameras could detect the large signs that typically gas stations have and read off the gas prices. The gas prices would then be disseminated and shared with other vehicles to collaboratively support a service that helps drivers find the cheapest gas around them.

While detecting available parking spots or reading off gas prices may be harder computationally than detecting a traffic signal, the time constraints are not as tight. Traffic signal detection needs to be fast so that new frames can be processed as frequently as possible and the traffic signal transition times are detected as accurately as possible. Particularly for the cheap gas advisory service, the time constraints are very loose. A windshield-mounted smartphone could capture an image and take several seconds or even minutes to process it and detect the prices. To improve the detection accuracy, still images could be captured instead of the video frames that we used to increase detection speed for SignalGuru.

10 Related Work

Several collaborative systems have been proposed that crowdsource information from GPS, accelerometer and proximity sensors in order to estimate traffic conditions [11], [21], [29], detect road abnormalities [7], collect information for available parking spots [20] and compute fuel efficient routes [8]. In [18] Lee *et al.* propose an application that lets police track the movement of suspicious vehicles based on information sensed by camera-equipped vehicles. Other works have also proposed to equip vehicles with specialized cameras and detect traffic signals with the ultimate goal of enabling autonomous driving [9], assisting the driver [22], or detecting the location of intersections and overlaying navigation information [28]. In [5], [31], the authors enforce traffic laws (*e.g.,* detection of red light runners) by detecting traffic signals and their current status with stationary cameras affixed to infrastructure. Furthermore, as we discussed in the introduction, approaches aiming to enable GLOSA have been based on costly infrastructure and hence failed to grow in scale. To the best of our knowledge, no other work has proposed to leverage commodity windshield-mount smartphone cameras, or above all, to *predict* the *future* schedule of traffic signals for the purpose of providing it to users and enabling the proposed set of novel applications.

Our camera-based traffic signal detection algorithm draws from several schemes mentioned above [9], [22], [28]. However, in contrast to these approaches that detect a single target, SignalGuru uses an iterative threshold-based approach for identifying valid traffic signal candidates. We also propose the IMU-based detection window scheme that leverages information from smartphones' accelerometer and gyro devices to narrow down the detection area offering significant performance improvements. In order to be able to detect ill-captured traffic signals under poor ambient light conditions, previous approaches either use normalized RGB images [22] or estimate ambient illumination [5]. In contrast to these approaches, we leverage the observation that LED traffic signals are a light source of a fixed luminous intensity, and provide mechanisms to perform a one-time automatic adjustment of the smartphone camera's exposure setting. In this way, the camera hardware is configured to capture traffic signal bulbs correctly regardless of the prevailing ambient light conditions, obviating the need for additional image processing steps. Last and most important, all these prior works focus solely on reporting the current status of traffic signals. They are not concerned with phase transitions and thus do not propose schemes to filter them, or collate the past traffic signal schedule for prediction of the future.

Services like SignalGuru that are based on collaborative sensing naturally have trust, privacy, security implications. SignalGuru can use DLT certificates [19] or a TPM [26] in order to improve trust in the exchange of traffic signal data. Furthermore, the SignalGuru-enabled devices and their users can be safeguarded by spatio-temporal cloaking [10] and other proposed approaches for grassroots participatory sensing [12].

11 Conclusions

In this paper, we presented a collaborative sensing platform that is based on the cameras of windshield-mounted smartphones. We discussed several novel services that this platform enables and focused on SignalGuru. SignalGuru is a novel service that leverages collaborative sensing on windshield-mount smartphones, in order to predict traffic signals' future schedule and support a set of novel applications in a fully distributed and grassroots approach. Our proposed schemes improve traffic signal detection, filter noisy traffic signal data, and predict traffic signal schedule. Our results, from two real world deployments in Cambridge (MA, USA) and Singapore, show that SignalGuru can effectively predict the schedule for not only pre-timed but also state of the art traffic-adaptive traffic signals. Furthermore, fuel efficiency measurements, on an actual city vehicle, highlight the significant fuel savings (20.3%) that our SignalGuru-based GLOSA application can offer. We hope that this work will motivate further research in our proposed collaborative sensing platform and the services that it can enable.

12 Acknowledgments

We acknowledge the support of NSF grant CSR-EHS-0615175 and the Singapore-MIT Alliance for Research and Technology Future Urban Mobility center. We would like to thank Chia-Hsin Owen Chen for his help in analyzing the traffic signal color distributions and Jason Gao for his help in adjusting the camera exposure time. We would also like to thank the volunteers at our two deployments (Prof. Ananda and Prof. Chan's groups at NUS and our group at MIT). Special thanks also to Singapore's Land Transport Authority ITS Centre and Cambridge's traffic engineer Jeff Parenti for their help in obtaining the traffic signal data and settings. Last but not least, our sincere thanks to the anonymous reviewers, our editor Marco Gruteser and our MobiSys shepherd Nigel Davies for their insightful comments.

References

[1] Behrang Asadi and Ardalan Vahidi. Predictive cruise control: Utilizing upcoming traffic signal information for improving fuel economy and reducing trip time. In *IEEE Transactions on Control Systems Technology*, 2011.

[2] Audi Travolution Project. http://www.audi.com/com/brand/en/tools/news/pool/2010/06/audi_travolution_.html.

[3] Roberto Baldessari, Bert Bödekker, Achim Brakemeier, et al. *Car 2 Car Communication Consortium Manifesto*, 2007.

[4] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[5] YC Chung, JM Wang, and SW Chen. Vision-based traffic light detection system at intersections. In *Journal of Taiwan Normal University: Mathematics, Science and Technology*, 2002.

[6] Gurcan Comert and Mecit Cetin. Queue length estimation from probe vehicle location and the impacts of sample size. *European Journal of Operational Research*, 2009.

[7] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, and Hari Balakrishnan. The pothole patrol: Using a mobile sensor network for road surface monitoring. In *MobiSys*, 2008.

[8] Raghu K. Ganti, Nam Pham, Hossein Ahmadi, Saurabh Nangia, and Tarek F. Abdelzaher. GreenGPS: A participatory sensing fuel-efficient maps application. In *MobiSys*, 2010.

[9] Jianwei Gong, Yanhua Jiang, Guangming Xiong, Chaohua Guan, Gang Tao, and Huiyan Chen. The recognition and tracking of traffic lights based on color segmentation and camshift for intelligent vehicles. In *IEEE Intelligent Vehicles Symposium (IV)*, 2010.

[10] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys*, 2003.

[11] Baik Hoh, Marco Gruteser, Ryan Herring, Jeff Ban, Daniel Work, Juan-Carlos Herrera, Alexandre M. Bayen, Murali Annavaram, and Quinn Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *MobiSys*, 2008.

[12] Baik Hoh, Marco Gruteser, Hui Xiong, and Ansaf Alrabady. Enhancing security and privacy in traffic-monitoring systems. *IEEE Pervasive Computing*, 2006.

[13] Gaurav S. Kasbekar, Yigal Bejerano, and Saswati Sarkar. Lifetime and coverage guarantees through distributed coordinate-free sensor activation. In *MobiCom*, 2009.

[14] Peter Koonce, Lee Rodegerdts, Lee Kevin, Shaun Quayle, Scott Beaird, Cade Braud, Jim Bonneson, Phil Tarnoff, and Tom Urbanik. *Traffic Signal Timing Manual*. U.S. Department of Transportation, 2008.

[15] Emmanouil Koukoumidis, Dimitrios Lymberopoulos, Karin Strauss, Jie Liu, and Doug Burger. Pocket cloudlets. In *ASPLOS*, 2011.

[16] Emmanouil Koukoumidis, Li-Shiuan Peh, and Margaret Martonosi. RegReS: Adaptively maintaining a target density of regional services in opportunistic vehicular networks. In *PerCom*, 2011.

[17] Emmanouil Koukoumidis, Li-Shiuan Peh, and Margaret Martonosi. SignalGuru: leveraging mobile phones for collaborative traffic signal schedule advisory. In *MobiSys*, 2011.

[18] Uichin Lee, Eugenio Magistretti, Mario Gerla, Bellavista, and Antonio Corradi. Dissemination and harvesting of urban data using vehicular sensor platforms. In *IEEE Transaction on Vehicular Technology*, 2008.

[19] Vincent Lenders, Emmanouil Koukoumidis, Pei Zhang, and Margaret Martonosi. Location-based trust for mobile user-generated content: applications, challenges and implementations. In *HotMobile*, 2008.

[20] Suhas Mathur, Tong Jin, Nikhil Kasturirangan, Janani Chandrasekaran, Wenzhi Xue, Marco Gruteser, and Wade Trappe. Parknet: Drive-by sensing of road-side parking statistics. In *MobiSys*, 2010.

[21] Prashanth Mohan, Venkata N. Padmanabhan, and Ramachandran Ramjee. Nericell: Using mobile smartphones for rich monitoring of road and traffic conditions. In *SenSys*, 2008.

[22] M. Omachi and S. Omachi. Traffic light detection with color and edge information. In *ICCSIT*, 2009.

[23] Openstreetmap. http://www.openstreetmap.org/.

[24] Sasank Reddy, Deborah Estrin, and Mani Srivastava. Recruitment framework for participatory sensing data collections. In *PERVASIVE*, 2010.

[25] RITA | ITS | Deployment Statistics Database. http://www.itsdeployment.its.dot.gov/Results.asp?rpt=M&filter=1&ID=320.

[26] Stefan Saroiu and Alec Wolman. I am a sensor, and I approve this message. In *HotMobile*, 2010.

[27] SCATS. http://www.scats.com.au/.

[28] Hwang Tae-Hyun, Joo In-Hak, and Cho Seong-Ik. Detection of traffic lights for vision-based car navigation system. In *Advances in Image and Video Technology*. Springer, 2006.

[29] Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Samuel Madden, Hari Balakrishnan, Sivan Toledo, and Jakob Eriksson. VTrack: Accurate, energy-aware road traffic delay estimation using mobile phones. In *SenSys*, 2009.

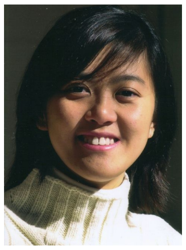
[30] J. van Leersum. Implementation of an advisory speed algorithm in TRANSYT. *Transportation Research Part A: General*, 1985.

[31] Yiyan Wang, Yuexian Zou, Hang Shi, and He Zhao. Video image vehicle detection system for signaled traffic intersection. In *International Conference on Hybrid Intelligent Systems (HIS)*, 2009.

[32] Pei Zhang and Margaret Martonosi. CA-TSL: Energy adaptation for targeted system lifetime in sparse mobile ad-hoc networks. *IEEE Transactions on Mobile Computing*, 2010.

Emmanouil Koukoumidis received his Dimploma in Electrical and Computer Engineering from the National Technical University of Athens and his master's degree in Electrical Engineering from Princeton University, where he is currently a PhD candidate. In the last two years, he has been a research scholar in MIT's Computer Science and Artificial Intelligence Labratory and Singapore's SMART Centre focusing on Intelligent Transportation Systems applications. His research interests include mobile computing, distributed systems and networking in general. He is member of the IEEE and the ACM.

Margaret Martonosi received the bachelor's degree from Cornell University and the master's and PhD degrees from Stanford University, all in electrical engineering. She is currently a professor of computer science at Princeton University, where she has been on the faculty since 1994. She also holds an affiliated faculty appointment in the Electrical Engineering Deparment at Princeton University. Her research interests include computer architecture and the hardware/software interface, with particular focus on power-efficient systems and mobile computing. She is a fellow of the IEEE and the ACM.



Li-Shiuan Peh is Associate Professor of Electrical Engineering and Computer Science at MIT since 2009. Previously, she was on the faculty of Princeton University from 2002. She graduated with a Ph.D. in Computer Science from Stanford University in 2001, and a B.S. in Computer Science from the National University of Singapore in 1995. Her research focuses on low-power on-chip networks, parallel computer architectures and mobile computing. She was awarded the CRA Anita Borg Early Career Award in 2007, Sloan Research Fellowship in 2006, and the NSF CAREER award in 2003. She is a member of the IEEE and the ACM.