Towards Safer Texting While Driving Through Stop Time Prediction

Hongyu Li[♮], Luyang Liu[♮], Cagdas Karatas[♮], Jian Liu[§], Marco Gruteser[♮]
Yingying Chen[§], Yan Wang[*], Richard P. Martin[♮], Jie Yang[◇]
[♮]WINLAB, Rutgers University, North Brunswick, NJ, USA
[♮]{hongyuli, luyang, cagdas, gruteser, rmartin}@winlab.rutgers.edu
[§]Stevens Institute of Technology, Hoboken, NJ, USA
[§]{jliu28, yingying.chen}@stevens.edu
[*]Binghamton University, Binghamton, NY, USA
[*]yanwang@binghamton.edu
[◇]Florida State University, Tallahassee, FL, USA
[◇]jie.yang@cs.fsu.edu

ABSTRACT

Driver distraction due to in-vehicle device use is an increasing concern and has led to national attention. We ask whether it is not more effective to channel the drivers' device and information system use into safer periods, rather than attempt a complete prohibition of mobile device use. This paper aims to start the discussion by examining the feasibility of automatically identifying safer periods for operating mobile devices. We propose a movement-based architecture design to identify relatively safe periods, estimate the duration and safety level of each period, and delay notifications until a safer period arrives. To further explore the feasibility of such a system architecture, we design and implement a prediction algorithm for one safe period, long traffic signal stops, that relies on crowd sourced position data. Simulations and experimental evaluation show that the system can achieve a low prediction error and its converge and prediction accuracy increase proportionally to the availability of the amount of crowd-sourced data.

CCS Concepts

•**Information systems** → **Mobile information processing systems;** •**Human-centered computing** → **Ubiquitous and mobile computing systems and tools;** *Empirical studies in ubiquitous and mobile computing;* •**Applied computing** → Transportation;

Keywords

Safety Driving, Smart Phone Application, Safety Aware Notification

CarSys'16, October 03-07, 2016, New York City, NY, USA
© 2016 ACM. ISBN 978-1-4503-4250-6/16/10...$15.00
DOI: http://dx.doi.org/10.1145/2980100.2980102

1. INTRODUCTION

Distractions due to frequent interruptions from mobile devices are an increasing concern [21]. This concern is most prominent where serious safety risks arise—for example when driving cars, handling construction equipment, or operating trains. In the United States, studies estimate that phone distractions contribute to 445 fatalities annually [20]. For this reason, laws in many states and countries do not allow handhold phone usage or texting while driving.

While these laws generally appear sensible, technology evolves quickly, which raises many questions about their effectiveness. First, wearable devices lead to uncertainty about the applicability of the laws and make enforcement of these laws evermore difficult, since it is often not clear whether a wearable device was actively used or simply worn [24]. Second, growing mobile device usage leads to increasing social pressure to respond to messages without delay. Third, cars make an increasing amount of information available through in-vehicle user, navigation, and entertainment interfaces.

Vehicle user interfaces are, of course, designed with the driver in mind and must follow specific guidelines [7]. Given the trend towards mobile operating systems for cars (e.g., Android Auto or Apple CarPlay) the boundary is becoming increasingly fuzzy. If such systems eventually allow third-party apps for cars, will all apps be carefully screened for distraction potential? Also, due to the longer life cycle of vehicles, outdated user interfaces and technology can be more distracting to use than up-to-date mobile devices.[1]

Given this increasing uncertainty, we ask whether it is not more effective to channel the driver's device and information system use into safer periods, rather than completely prohibiting handheld mobile device use. The need for driver attention is highly variable during most trips, which means that there are periods where device use can be relatively safe. The safest period is likely when the vehicle is stopped, which frequently occurs at traffic lights or occasionally during traffic congestion. The trend towards automated driving can be

[1]Compare, for example, the time needed to enter and find a navigation destination on an older built-in car navigation system with the time needed on a modern phone navigation app. Of course, this function was never intended to be used while driving but this is unlikely to hold in practice.

expected to create additional periods that are relatively safe. While current production systems still require the driver to supervise the system, technology road-maps foresee that the vehicles will eventually be able to drive parts of a trip without driver supervision. Could we achieve an overall gain in safety if the mobile system can steer potential interactions into such periods and away from more dangerous periods?

A conclusive answer to this question, however, also requires additional consideration of human behavioral, ethical, and legal dimensions. More generally, whether mobile operating systems should manage user attention has been asked by Garlan [8] and more recently by Lee and colleagues [16]. To date, however, no unifying approach for estimating the user capacity for attention has emerged. We argue therefore that development should start with the most pressing need, in particular the driving context, and attempt to move the discussion to a more concrete technical proposal.

Therefore, this paper aims to start this discussion by examining the technical feasibility and by proposing a technical approach realized by stop time prediction. In particular, we propose a movement-based system that identifies relatively safe periods, estimate the duration and safety level of each period, and delay notifications until a safer period arrives. To demonstrate the feasibility of this approach, we develop an algorithm that predicts the frequently occurred traffic light waiting periods to determine the safe periods. Our algorithm can be deployed in a crowdsourcing environment, the performance increases as the availability of the crowd-sourced data increases. Our system shows the possibility of determining the safe periods through stop time prediction by using the limited crowd-sourced traffic data.

2. RELATED WORK

Reducing phone distractions while driving has been the subject of much previous research.

Reducing Interactions while Driving. The most direct way to eliminate the distraction from drivers' phones is to completely ignore incoming calls and texts. For instance, quite a few smartphone applications, e.g., *OneTap* [4], *AT&T DriveMode* [2], and *Live2Txt* [3], use GPS information to automatically sense when a user is driving and suppress all the notifications that could distract the drive from the road. Instead of using GPS information, *Agent* [1] assumes users are driving when their smartphones are connected to vehicles' Bluetooth systems, and automatically responds a message to people who call or text the drivers with their driving status. There are strict approaches that utilize additional hardware installed in vehicles to completely interfere cellular signals [5, 23] to eliminate distractions from phone usage while driving. Lindqvist and Hong [17] move a step further to discuss how drivers' phones could interact with callers by context-awareness. They propose to let callers know callees' driving status and allows the callers to decide whether to call or text the callees later. In order to obtain people's driving status, a variety of approaches have been proposed to distinguish drivers' phone from passengers [26, 25, 18]. However, all these papers seek to reduce drivers' interactions with their phones during driving without considering the context and human behavior.

Intelligent Notification. Some researches have focused on managing smartphone notifications intelligently. Horvitz et al. [12] propose to apply the Bayesian model to infer users' available attention level and compute expected costs of interruptions and deferring alerts based on calender information. However, the system is limited by the Bayesian model itself, namely whether there is a strong relationship between current and previous contexts. Ho and S.Intille [11] find that there are potential self-initiated task interruptions that lie in physical activity changes, such as the time when a user's status transits from sitting to standing may indicate that the user is taking a break, which might be the best time to push a reminder notification. However, such transitions of human activities and their context are hard to detect with only mobile devices. Mehrota et al. [19] construct a machine-learning model to predict the acceptance of a notification by analyzing its content and context Lee et al. [16] argue that users' attention should be managed as a resource that should be considered when evaluating every potential interaction, in addition to its importance. However, all these papers mentioned above only consider general notification contexts, which may not fit into real-life scenarios with significant safety concerns, such as smart phone notification while driving.

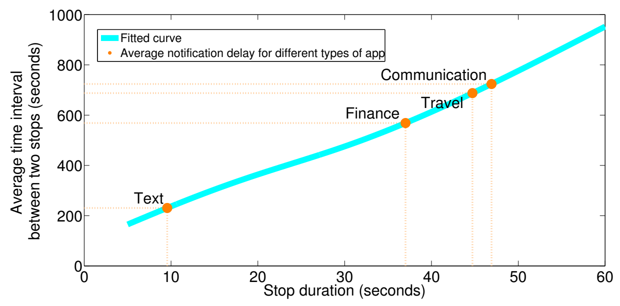


Figure 1: Average time interval between stops with respect to stop duration.

3. CHANNELING DRIVER ATTENTION

Using mobile devices when driving is often viewed as a dangerous behavior that might lead to accidents. While much work aims to avoid driving distraction by blocking notifications from their mobile devices, we do observe the existence of relatively safer periods that drivers can use to operate their mobile devices, such as while stopped at a traffic light, blocked in traffic congestion, or waiting at a gas station, etc. What if mobile devices were able to channel the drivers' attention to devices into such relatively safer periods, instead of completely blocking notifications of driver's mobile devices while driving? To answer this question, let us first explore the answers of the following three questions.

How common are safe opportunities to interact with information systems during driving? The safest opportunities to interact with devices are when the vehicle is stopped. We therefore analyze typical stop times during trips using a public data set [13] of driver telematic traces from an European insurance company[9]. This data set contains 547,200 trips from 50,000 anonymized drivers, and the average duration of each trip is 10.85 minutes. We processed all the trips and show the length of stop intervals with respect to a given stop duration in Fig 1. For example, a stop longer than 9.56s (which is the minimum time requirement to reply a short text message, details discussed in the next paragraph) occurs every 3.96 minutes. Thus, for a 10min trip, one would expect about 2.5 such opportunities.

Many of these safe opportunities occur when vehicles are

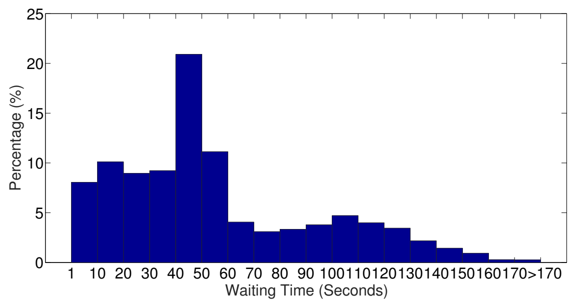


Figure 2: The distribution of waiting time.

waiting at traffic lights. Since the prior data set does not provide any information about the reason for each stop, we analyze traffic signal waiting time data separately from an adaptive traffic signal system on a segment of U.S. Route 1 in New Jersey, which parallels the east coast of the United States and is an important part of commute routes of many local residents. This data set obtained from the NJ Department of Transportation (NJDOT) consists of the records of 22 intersections along U.S. Route 1 over the recent 9 months and includes the waiting time of the first car in queues. As shown in Fig 2, in 92% of cases the first car's waiting duration is more than 10s, and in 32% of cases it exceeds 60s.

How much time is needed for short interactions? Experiments show that people only need 9.56s to text a 10 characters message [10]. In real life conditions, the average interaction duration with finance applications (such as Bank of America, Google Finance, iStockManager), travel applications (e.g., Google Maps, Yelp and Waze) and communication applications (e.g., Google Mail, Handcent SMS, K-9 Mail) on an Android phone is 37.01s, 44.72s, 46.92s respectively [6].

How much would interactions be delayed? The system could channel driver interaction with mobile information systems into such short interactions when the vehicle is stopped by delaying notifications and incoming messages until such an opportunity arises. The time interval between two different stops of suitable duration is the upper bound of for the arising interaction latency. Fig 1 shows the latency of four different kinds of interactions, which we discussed in the second question. As can be seen from Fig 1, this would result in 230.54s latency to reply a text message. For finance, travel and communication applications, we need at most 568.64s, 687.58s and 723.95s until we encounter a suitable stop. Therefore, the analysis of data sets shows that we could expect multiple safe chances to interact with information systems within a reasonable delay.

4. SYSTEM DESIGN

Next we discuss the architecture design of the system and illustrate the feasibility of the design by showing an example system under the context of the driver waiting at traffic lights.

4.1 System Architecture

4.1.1 Overview

We propose a system that channels the driver's devices usage into safer time periods rather than a complete prohibition on some devices and unregulated use of others while driving. The main goal of the system is to steer potential interaction into safe time periods (i.e., when vehicles are stationary) by delaying the notifications from various applications (e.g. the incoming text or chat messages), until a sufficiently long safety period arises. The system focuses on notifications that can distract and encourage a driver to interact with the device, but does not prevent drivers from starting interactions. We envision that the system would address a significant number of interactions that lead to drivers' distractions.

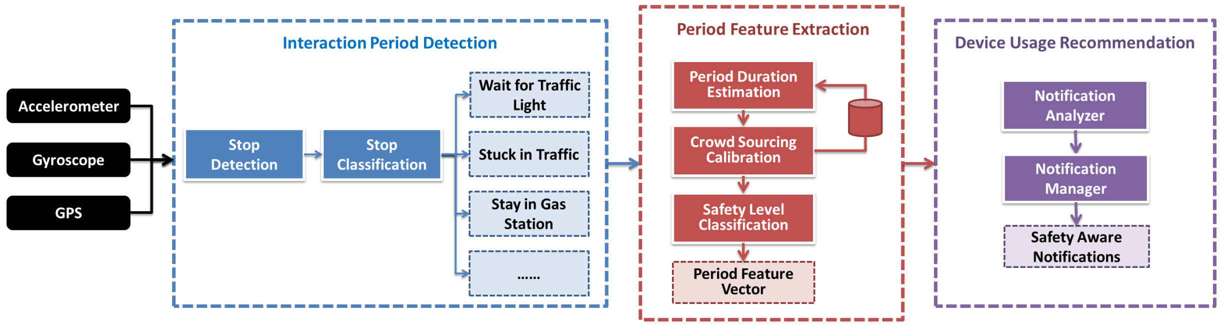
Based on the heuristic that the risk of a driver's interaction with the device is at its lowest when the vehicle is stationary, the system could utilize the vehicle's stop periods as the potential safe interaction periods. A movement-based approach can be used to detect vehicles' stop events by leveraging inertial motion sensors of mobile devices, such as accelerometers and gyroscopes. Such an approach could provide high stop detection accuracy at relatively low power consumption. The system could then predict the stop duration for each event and further improve its prediction accuracy by crowd sourcing driving behavior from other drivers. By also considering the context of each stop, the system could derive how safety the current stop is, and output it together with predicted stop duration as a feature vector. Finally, the system schedules the notifications and only releases the notifications that can be interacted during safer stop duration. Notification prioritization, interaction friendly delivery channels should be also considered when pushing notifications to the driver.

Fig. 3 illustrates the architecture design of the system. *Interaction Period Detection* module is responsible to detect and classify the stop periods through inertial and GPS data. For a detected stop period, *Period Feature Extraction* module will be able to extract the period feature vector, which will be the input of *Device Usage Recommendation* module to channel safer notifications. We note that, to steer interactions into safe periods, the mobile device should firstly be aware that their owner is driving. Existing work [26, 25, 18] have solutions to distinguish between drivers and passengers. Although these techniques are not perfectly accurate, a false positive would result in only a small delay in a notification, which is usually tolerable, and a false negative would simply lead to a notification being delivered while driving, as is the case right now.

4.1.2 Potential Interaction Period Detection

As potential interaction periods usually happen when vehicles are stationary, the system devises a *Stop Detection* module using inertial data to detect vehicle stops. Compared to GPS based approaches, inertial sensors in mobile devices have relatively lower power consumption. Once a potential stop is detected, the *Stop Classification* module requests for an accurate GPS location to further differentiate different types of stops (e.g., waiting for traffic lights, stuck in traffic jams and staying in gas stations) based on the location context of the stops. For example, the driving trajectory and corresponding map information, as the location context of stops, could be used to distinguish whether the vehicle is on or off the road and whether it is at a point of interest such as a gas station. If the vehicle stops on the road where the map indicates traffic congestion, the driver is probably stuck in a traffic jam.

4.1.3 Period Feature Extraction



Accelerometer
Gyroscope
GPS
Interaction Period Detection
Stop Detection
Stop Classification
Wait for Traffic Light
Stuck in Traffic
Stay in Gas Station
......
Period Feature Extraction
Period Duration Estimation
Crowd Sourcing Calibration
Safety Level Classification
Period Feature Vector
Device Usage Recommendation
Notification Analyzer
Notification Manager
Safety Aware Notifications

Figure 3: Architecture Design.

We find that the types of stops provide two important features for determining the safe period for interaction: period duration and safety levels of stops. Given a specific type of stop, the system should be able to predict the stop duration through the *Period Duration Estimation* module. In the case of waiting for traffic light, the stop duration could be predicted by current traffic phase schedule and a traffic queue discharge model, which will be discussed in Section 4.2. If a driver got stuck in traffic, we could make conservative predictions based on the frequency he stops during recent period, with the intuition that the heavier traffic will cause less frequent but longer duration stops. For staying in gas station, the historical time to fill up the vehicle and the popularity of the gas station could be used to estimate the stop duration. Additionally, the *Crowd Sourcing Calibration* module is devised to improve the prediction accuracy and predict future trends by exploiting the real time parameters extracted from other drivers' devices, such as the traffic schedule during waiting for traffic light, the traffic condition when drivers stuck in traffic and the gas station popularity if someone stops for gas refill.

The *Safety Level Classification* module derives another period feature, safety level, to describe how safe the each stop period would be for in-vehicle device interaction. The safety level mainly depends on the type of the stop, such as staying in a gas station would provide higher level of safety for interaction than that of stuck in a traffic jam. For the same type of stops, the system could also fine-tune the safety level. For example, the heavier the traffic congestion is, the higher the safety level of the stop faces. After obtaining the safety levels, a period feature vector is built for each stop to help determine the right safety period for device interaction during driving.

4.1.4 *Device Usage Recommendation*

After determining stop periods and their safety level, our system can intelligently deliver appropriate notifications, which can be handled within the estimated stop period through user friendly interaction channels in safe periods. Towards this end, the *Notification Analyzer* can extract the characteristics of notifications, such as the urgency of notifications and users' expected interaction time with notifications. Based on the characteristics of notifications, *Notification Manager* will be responsible for orchestrating notifications such as blocking the notifications and forwarding to appropriate delivery channels(i.e., screen displays,audio alarms, head-up displays) in a timely and non-distracting manner.

4.2 Case Study: Traffic Light Waiting Time Prediction

Waiting for a traffic light provides the most potential for drivers to interact with in vehicle devices, since it enables more frequent and longer stop periods. Therefore, we implement an example system under the context of waiting for traffic lights by extending the architecture design in Section 4.1. Specifically, we realize the period duration estimation to estimate the traffic light waiting duration through predicted traffic schedules based on previous work of traffic schedule prediction [15]. Since the traffic schedule is not invariant, we further propose a crowd sourcing calibration mechanism, which could not only fast converge the prediction accuracy based on few samples, but also increase the prediction performance with proportional to more samples.

4.2.1 *Implementation Overview*

Based on the architecture shown in Fig 3, we extend and implement an example system under the context of waiting for traffic lights. Fig. 4 illustrates the details of the expanded blocks. In the example system, we mainly focus on the extension of Period Duration Estimation and Crowd Sourcing Calibration modules under the context of traffic light waiting time prediction. Other cases distinguished by the Stop Classification module could have their own detail design and implementation for the Period Duration Estimation and Crowd Sourcing Calibration modules. The modules with slash shadows in Fig. 4 will not be implemented and discussed in this case study.

The system work flow starts with *Interaction Period Detection* module as shown in Fig. 4, which is responsible for (1) detecting when the driver stops through inertial and GPS data, (2) determining which traffic light the vehicle is waiting for based on driving direction and map information. Once a stop event is detected, the system starts to estimate the vehicle waiting duration, which includes two parts: waiting time before the traffic light turning to green and waiting time for vehicle queue discharging. To estimate these two waiting times, the system performs the *Waiting Time Prediction* and *Queue Discharge Model*, respectively. When the vehicle moves, the ground truth of waiting time prediction is fed back to the system, and the *Sample Pool Update* module is triggered to add the new sample of waiting time into the sample pool. The system further calculates the error of predicted waiting duration in the *Error Checking* and decide whether to perform the *Sample Pool reconstruct* according to the most recent sample. After updating the sample pool, the system recalculates the cycle length and offset through the

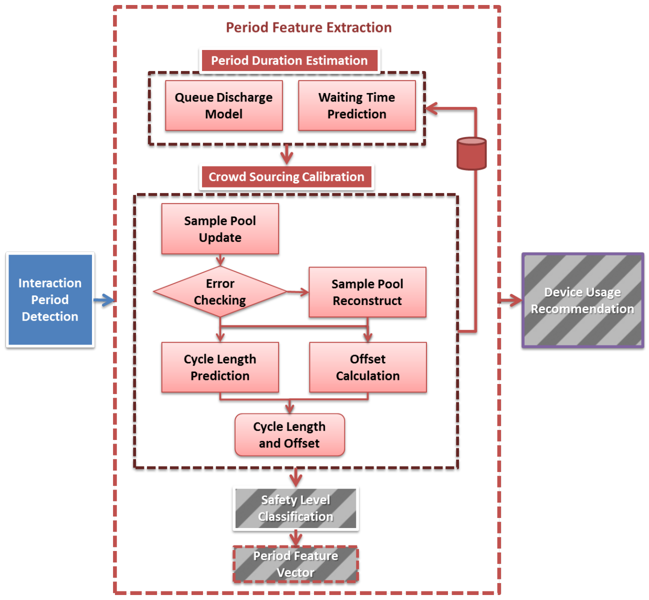


Figure 4: Wait for traffic light case study.

Cycle Length Prediction and *Offset Calculation*. The output would be the latest predicted set of *Cycle Length and Offset*, which will be used to the next waiting time prediction.

4.2.2 Traffic Schedule Prediction Background

To predict the traffic light waiting time, we estimate the waiting duration based on predicted traffic light schedule. The schedule for a specific direction of an intersection could be modeled with the length of cycle. Without loss of generality, a cycle is defined as a complete sequence of signal indications, which starts with a green phase followed by a yellow phase and a red phase. To simplify the model, the yellow phase is considered the same as the red phase. Thus the length of cycle is the time duration of a complete cycle.

There are two kinds of traffic light systems with different scheduling of traffic lights. The adaptive traffic light system automatically changes the length of cycles according to current traffic conditions, while the pre-timed traffic light system uses predefined length of cycle in different time spans of a day. Our system is designed to be able to work for both these two kinds of traffic light systems.

In this work, we predict the length of cycle based on Kerper et al.'s work [15], which keeps optimizing the estimated cycle length with respect to the resulting error in the offset. We search the cycle length (x) in a specific space ($\overrightarrow{X}$), through minimizing the mean squared error (MMSE) of offset calculations as shown in Formula 1:

$$\begin{aligned} &\underset{x\in\overrightarrow{X}}{\operatorname{argmin}} \sum_{i=1}^{m}(dt_i - \bar{dt})^2 \\ &dt_i = t_{\text{cycleStarti}} \; mod \; x, \qquad (1) \\ &\bar{dt} = (1/m) * \sum_{j=1}^{m} dt_j, \end{aligned}$$

where $t_{\text{cycleStarti}}$ is the time when a cycle starts (i.e., when the traffic light changes to green), dt_i is the offset with respect to current cycle length (x) and $t_{\text{cycleStarti}}$ and $\bar{dt}$ is the average offset calculated by all $t_{\text{cycleStarti}}$ samples. The search space of cycle length in our system is from 50s to 200s, which covers most range of cycle length in real traffic light systems.

4.2.3 Traffic Light Waiting Duration Estimation

In order to predict the traffic light waiting time, the system use *Queue Discharge Model* to predict how long the vehicle takes to leave the queue, and perform *Waiting Time Prediction* to estimate how long the vehicle would wait until the traffic light changes to green phase.

Algorithm 1 Traffic Light Waiting Duration Estimation

```
Input: stop event, cycle length, offset
Output: estimated stop duration
 1: function DurationEstimation(stop event, cycle length, offset)
 2:     add stop event into the global queue;
 3:     if $\text{Pool}_{\text{all}}$.size>=2 then
 4:         $T_{\text{GreentoLeave}} \leftarrow \text{QModel}(\text{distance})$;
 5:         $T_{\text{StoptoGreen}} \leftarrow \text{DPredict}(\text{cycle length}, \text{offset})$;
 6:         $T_{\text{waiting}} \leftarrow T_{\text{StoptoGreen}} + T_{\text{GreentoLeave}}$;
 7:         return $\text{T}_{\text{waiting}}$;
 8:     else
 9:         return null;
10:     end if
11: end function
```

The detailed process of duration estimation is shown in Algorithm 1. We assume the system maintains a global stop event queue that collects all the stop events occurring during the current cycle. Each entry in the queue is a stop event that records the time when the vehicle stops and the distance between vehicle's stop location to the center of the intersection. Our algorithm takes a stop event, and a set of cycle length and offset predicted by calibration module as the input to estimate the traffic light waiting duration. (Notice that, the set of cycle length and offset could be empty if the system has not make the first guess.) As a starting point, the algorithm put the input stop event into the global stop event queue in Line 2. $Pool_{\text{all}}$ is introduced to store the cycles of data samples which could be used to predict latest cycle length and offset, each element of $Pool_{\text{all}}$ is a sub-pool which stores all the data samples within one cycle. Thus the size checking of $Pool_{\text{all}}$ in Line 3 makes sure there are at least 2 cycles of samples could be used to perform the estimation, which means the calibration module should have already make a prediction for the cycle length and offset. If there are enough sample of stop events in the $Pool_{\text{all}}$, the algorithm predicts the waiting time when the traffic light turning to green phase ($T_{StoptoGreen}$) and the waiting time for vehicle queue discharging ($T_{GreentoLeave}$), and calculate the estimated waiting duration ($T_{waiting}$) for the stop event through Line 4 to 6. To predict the duration that a vehicle takes to leave the waiting queue, we use a simple linear queue discharge model [22] to get the waiting duration based on vehicle's stop distance which implemented by QModel(distance). The duration of waiting for traffic light turning to green could be predicted by Equation 2 which implemented as DPredict(cycle length, offset):

$$T_{\text{StoptoGreen}} = mod((t_{\text{stop}} - \text{Offset}), \text{Cycle Length}), \qquad (2)$$

where t_{stop} is the time when the vehicle actually stops, variables Offset and Cycle Length are inputs that updated by the Crowd Sourcing Calibration.

4.2.4 *Crowd Sourcing Calibration Mechanism*

The schedule of traffic systems are not invariable, even for pre-timed traffic systems, the cycle length varies due to pre-set poliiey or pedestrian query. Thus, the system should not only be able to calibrate and converge its prediction with latest samples, but also improve its prediction accuracy proportional to the number of crowd sourced data samples.

Algorithm 2 Crowd Sourcing based Calibration

```
Input: stop event, groundtruth duration of this stop
 1: function CALIBRATION($stopevent, t_{\text{stopduration}}$);
 2:     estimate $t_{\text{green}}$;
 3:     add $t_{\text{green}}$ to sub-pool;
 4:     errorArray.add(error);
 5:     if have received ground truth of all stop events in the
        global queue then
 6:         POOLUPDATE($\text{Pool}_{\text{all}}$, sub-pool);
 7:         if $\text{Pool}_{\text{all}}$.size<=1 then
 8:             return ;
 9:         else if ERRORCHECKING(threshold) then
10:             POOLRECONSTRUCT($\text{Pool}_{\text{all}}$);
11:         end if
12:         cycle length ← CLPREDICT($Pool_{\text{all}}$);
13:         offset ← OFFCALCULATE($Pool_{\text{all}}$, cycle length);
14:         Initialize sub-pool and errorArray;
15:     end if
16: end function
```

The work flow of our fast converge and performance proportional calibration mechanism is described in Algorithm 2. The algorithm is called when a previously stopped vehicle moves, thus the ground truth stop duration could be obtained by the duration between when the vehicle stops and when the vehicle moves. Based on the ground truth of the stop duration, the time when the traffic light turning to green (t_{green}) could be calculated through Equation 3:

$$t_{\text{green}} = t_{\text{stop}} + t_{\text{stopduration}} - T_{\text{GreentoLeave}}, \qquad (3)$$

where t_{stop} is the time when the vehicle stops and $t_{stopduration}$ is the ground truth of the stop duration. Since t_{green} is the start point of a cycle, sets of t_{green} could be used as $t_{\text{cycleStarti}}$ in Equation 1 to predict the cycle length. All the t_{green} derived from the samples within current cycle would be stored into a sub-pool, which make sure to include all samples within one cycle to decrease the noise from Queue Discharge model for future prediction and contribute to increase the prediction accuracy proportional to the number of samples. In addition, the absolute prediction error of current stop event should also be stored into the errorArray for future condition checking.

The cycle length re-prediction will only be triggered if the ground truth of all stop events in the queue has been received, which indicates that the system have received the feedback of all the stop events in the current cycle. We note that this condition does not hold for cumulative traffic waiting queues, but it could be solved by using a clustering method or a maximum queue life cycle. Starting from Line 6 in Algorithm 2, the sub-pool containing the t_{green} will be added into Pool_{all} as an entry through calling POOLUPDATE(Pool_{all}, sub-pool). Before the system evaluating the change of schedule, the size of Pool_{all} is checked to distinguish whether the system has two or more cycles' data to predict the length of cycle. If the size of Pool_{all} is no larger than 1, the algorithm terminates without updating. Otherwise, the system evaluates the change of schedule by checking whether the mean value of errorArray is bigger than a threshold through ERRORCHECKING(threshold). In case the mean error is larger than the threshold, indicating that a schedule change occurs in the current cycle, the algorithm replaces the Pool_{all} with the subpools of the most recent two cycles by calling POOLRECONSTRUCT(Pool_{all}). This step guarantees that the calibration mechanism could converge faster than general mean filters, since the future predictions are based on most recent data. Finally, the system predicts the new length of cycle and offset through CLPREDICT(Pool_{all}) and OFFCALCULATE(Pool_{all},cycle length) based on the updated $Pool_{all}$, and initialize the sub-pool and errorArray. The updated length of cycle and offset are used for future waiting time duration estimation in Algorithm 1.

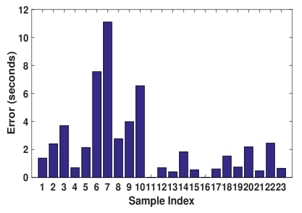
5. PERFORMANCE EVALUATION

In this section, we present the performance of our system in the case of traffic light waiting time prediction. Our system is implemented as discussed in Section 4.2, and we first evaluate the system using practical data we collected while driving on real road. To further explore the prediction accuracy on different kinds of traffic systems and the data requirements of how many data samples we need from crowd sourcing, we also evaluate the system based on simulation data. We use the prediction error, which is defined as the absolute value of the difference between prediction waiting time and ground truth waiting time, to evaluate the system. If there is more than one sample within a cycle, we evaluate the system by the mean error of all the samples inside each cycle.

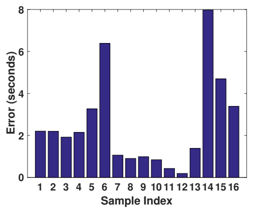
5.1 Evaluation based on Practical Data

For the practical data collection, we have two drivers driving around the intersection of Lincoln Hwy and Plainfield Ave in Edison of New Jersey (USA) in rush hours of two different days. In the first day, we collect 25 stops between 3 : 30 PM to 4 : 15 PM, and we further collect 18 stops between 5 : 15 PM to 6 : 00 PM in the second day. In total, there are 43 stops for a specific driving direction with a 61.43s average stop duration. During the data collection, one of the drivers drives a Toyota Camry with a Nexus 5 and the other driver drives a Honda Civic with a Nexus 6. The data are collected by an Android APP, which implements the stop detection module as discussed in Section 4.1.2. The rest of system are implemented offline on a server to process the data. The traffic light schedule system in this intersection is pre-timed, which means the cycle length does not automatically change based on real time traffic.

The prediction accuracy is shown in Fig 5. Fig 5a shows the prediction error for the first day, and the mean error is 2.37s. Fig 5b shows the prediction error of the second day with the mean error 2.50s. There are several peaks in both figures, those error bars are caused either by the noise of queue discharge model or the offset changes made by pedestrian request. The prediction results demonstrate that our system could accurately predict waiting time of traffic lights enabling the opportunities of safer texting while driving.



(a) Evaluation on the first day data.



(b) Evaluation on the second day data.

Figure 5: Prediction error on experimental data.

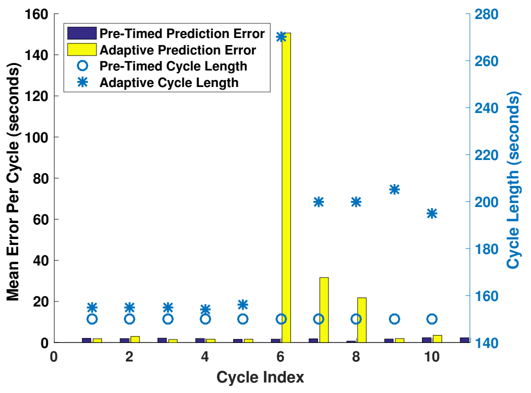


Figure 6: Prediction error of different traffic systems.

5.2 Evaluation based on Simulation Data

To further explore the performance of our system, we evaluate the system relying on simulation data by two folds. Firstly, we simulate both pre-timed and adaptive traffic systems to evaluate the waiting time prediction accuracy of our system. Secondly, we try to check the data requirement of our system to understand how the number of cycles and the number of samples inside one cycle affect our system performance.

5.2.1 *Prediction Accuracy Evaluation*

In order to simulate the real traffic conditions, we sample the traffic volume n_i (the number of vehicles waiting for a red light phase) from Poisson distribution. The λ_i of the Poisson distribution is sampled from an Uniform distribution of the range [1, 30], which covers the traffic volume range from off peak hour to rush hour for most intersections. To simulate vehicles queue discharge behavior, we use the same model as introduced in Section 4.2.2 and add a Gaussian noise with 2s variance on it. Besides, we use two different ways to simulate different traffic control systems. To simulate pre-timed traffic light system, we just manually specify the cycle length and green phase length. For simulating adaptive traffic light system, we use real adaptive system cycle length and green phase length obtained from NJDOT. During the experiment, the simulation data generation program will feed the system for 10 cycles data with two dimensional vectors (i.e., [the time when a vehicle stopped, the distance between the stop position and the intersection]), and the time when the vehicle moves after the current red light phase. The average stop duration of all the simulation data is 44.02s.

Th prediction errors in our simulation sre shown by bars in Fig 6 corresponding to left y axis, and the cycle length ground truth used to generate simulation data is shown in right y axis. The system only has a 1.43s mean prediction error for the pre-timed traffic system data set, because the cycle length does not change during these cycles. For the adaptive traffic system simulation, the mean prediction error increases to 14.51s, since the cycle length is not stable in this scenario. During the first five cycles, the cycle lengths vary in a small range, therefore there is not much error caused by the variation of cycle lengths. However, there is a tremendous change of cycle length happened in the sixth cycle, which causes the highest error bar. Actually, the system tries to rebuild the sample pool and predict the new cycle length based on the samples from 6th, 7th and 8th cycle. After this process, the system prediction accuracy converge quickly and the mean error decreases into small stable values again in the last two cycles. From Fig 6, we find that (1) the system prediction only has errors less than 3s with either stable or small variant cycle lengths. (2) Tremendous changes of cycle lengths result in large errors, but the prediction accuracy of our system converges quickly and the errors decrease within two cycles after the such changes.

5.2.2 *Data Requirement Evaluation*

We also evaluate our system in terms of how many data samples that we need from crowd sourcing. The number of data samples the system uses has two parameters: one is the number of cycles we have, the other is the number of samples within one cycle. Here, we use the same simulation method mentioned in Section 5.2.1 to generate three hundreds of data sets by specifying three different numbers of samples within one cycle (Each number of samples within one cycle generate 100 data sets). Instead of making the cycle length fixed or changing, we add a 2.2s variance (which is the average cycle length variance of the adaptive traffic system on Route 1 in New Jersey) on cycle length to archive a median variant case of traffic systems. And the average stop duration of the simulation data is 31.14s.

The results are shown in Fig 7. From the perspective of how the number of cycles affects our prediction accuracy, we could observe that starting from the 3rd cycle, the system could perform prediction, and the prediction error decreases as the number of cycle increasing. After five cycles, all of the one sample per cycle, three samples per cycle and five samples per cycle data set prediction could archive the accuracy less than 4.3s. It is because that the more cycles we have, the more accurately we could predict the cycle length, so that all of these three curves converges as the number of cycle increased. But the error is not stable because of the variance on the cycle length and the noise on queue discharge model. In terms of how does the number of samples within one cycle affect the system performance, we find on the figure that the more sample we have within one cycle, the more accuracy we could archive. It is because that more samples within one cycle could be used by the crowd sourcing calibration module to decrease the noise from queue discharge model. Therefore, based on the simulation data set, our system could perform predictions within 4.3s error after five cycles' data, and the prediction accuracy is performance proportional with the number of cycles and the number of data samples with one cycle.

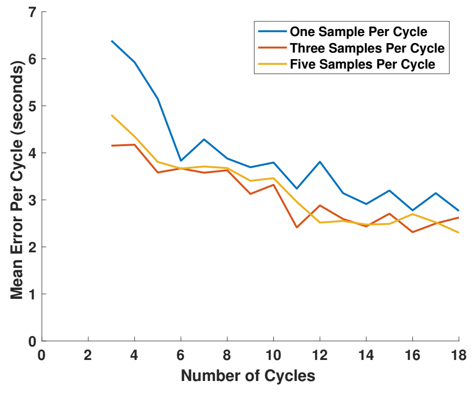


Figure 7: Prediction error varies with the number of cycles and number of samples within one cycle.

6. CONCLUSION AND DISCUSSION

In this paper we explore the possibility of improving the safety of drivers' interactions with mobile devices by limiting the interaction periods to relatively safe ones, in particular when the vehicle is stopped. Preliminary vehicular trip measurements and smart phone usage show that there are sufficient stopping periods in most trips to enable short, text based communication during these periods. To demonstrate the feasibility, we propose an architecture design and further implement parts of it under the case of waiting for traffic lights. The system performs an accurate waiting duration prediction based on both practical and simulation data sets. Through both driving experiments on real roads and simulations, we find that the prediction of the current vehicle's waiting time at a traffic light depends on the availability of the data samples of two vehicles ahead of it in the previous two traffic light cycles, indicating only limited crowd sourced data is needed to facilitate the prediction. Our fast converge and performance proportional calibration mechanism enable the system to correct itself through the increased amount of the crowd-sourced data. Our results provide confidence to channel safer notifications while driving.

We further discuss the future research direction focusing on the hardware and software extension of our system. Towards the hardware end, enabling traffic infrastructures (e.g., the signal controllers of each intersections) to communicate their states and action schedules with smart phones is a key way to improve our approach. Although many dedicated short range communication (DSRC) [14] systems have been proposed for traffic applications, there is still no uniformed public interface to feed drivers with real time traffic information (e.g., traffic schedules and traffic volumes at intersection level). The development of smart intersection signal controllers and their interfaces are still open research topics that can facilitate our system. On the software side, we will focus on the implementation of our safety level classification and crowd sourcing mechanism. To quantify the safety level for each stop event, it is necessary to analyze the mapping relationship between the types of stop events and drivers' attention to enable safe driving. In addition, we will need to design rules in our crowd sourcing mechanism to enable robust classifications of the types of stop events, since there are more complicated causes for stop events in real world.

Acknowledgment

This material is based in part upon work supported by the National Science Foundation under Grant Nos. CNS-1329939, CNS-1409811, CNS-1409767 and CNS-1505175.

7. REFERENCES

[1] Agent. https://play.google.com/store/apps/details?id=com.tryagent.
[2] AT&T DriveMode . https://play.google.com/store/apps/details?id=com.drivemode&hl=en.
[3] Live2Txt . https://play.google.com/store/apps/details?id=com.call.disconnect&hl=en.
[4] OneTap . http://www.getonetap.com/.
[5] Text Blocker . http://www.scosche.com/cellcontrol-safe-driving-system-for-cell-phones.
[6] Böhmer et al. Falling asleep with angry birds, facebook and kindle: a large scale study on mobile application usage. In *HCI*, 2011.
[7] Driver Focus-Telematics Working Group and others. Statement of principles, criteria and verification procedures on driver interactions with advanced in-vehicle information and communication systems. *Alliance of Automotive Manufacturers*, 2006.
[8] Garlan et al. Project aura: Toward distraction-free pervasive computing. *Pervasive Computing*, 2002.
[9] J. Goodman and P. Moreton. Axa: the global insurance company. *Harvard Business School Case*, (9-793):094, 1995.
[10] He et al. Mutual interferences of driving and texting performance. In *HFES*, 2014.
[11] J. Ho and S. S. Intille. Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In *CHI*, 2005.
[12] Horvitz et al. Bayesphone: Precomputation of context-sensitive policies for inquiry and action in mobile devices. In *User Modeling 2005*.
[13] Kaggle. Driver Telematics Analysis . https://www.kaggle.com/c/axa-driver-telematics-analysis.
[14] J. B. Kenney. Dedicated short-range communications (dsrc) standards in the united states. *Proceedings of the IEEE*, 99(7):1162–1182, 2011.
[15] M. Kerper, C. Wewetzer, A. Sasse, and M. Mauve. Learning traffic light phase schedules from velocity profiles in the cloud. In *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*, pages 1–5. IEEE, 2012.
[16] K. Lee, J. Flinn, and B. Noble. The case for operating system management of user attention. In *HotMobile*, 2015.
[17] J. Lindqvist and J. Hong. Undistracted driving: a mobile phone that doesn't distract. In *HotMobile*, 2011.
[18] Liu et al. Toward detection of unsafe driving with wearables. In *WearSys*, 2015.
[19] Mehrotra et al. Designing content-driven intelligent notification mechanisms for mobile applications. In *Ubicomp*, 2015.
[20] U. NHTSA. Distracted driving 2013. *Traffic Safety Facts Research Note*, 2013.
[21] W. H. Organization et al. Mobile phone use: a growing problem of driver distraction. 2011.
[22] N. Rouphail, A. Tarko, and J. Li. Traffic flow at signalized intersections. In *In Revised Monograph on Traffic Flow Theory, Chapter 9*. Citeseer, 1992.
[23] H. A. Shabeer, R. W. Banu, and H. A. Zubar. Technology to prevent mobile phone accidents. *IJENM*, 2012.
[24] U. Today. Woman fights ticket for driving with google glass, 2013.
[25] Wang et al. Sensing vehicle dynamics for determining driver phone use. In *MobiSys*, 2013.
[26] Yang et al. Detecting driver phone use leveraging car speakers. In *MobiCom*, 2011.