# GreenDrive: A Comprehensive System to Optimize Driving speed

Yiran Zhao[1], Tianyuan Liu[2], Yang Zhang[2]
1. Department of Electronic Engineering, Shanghai Jiao Tong University, China
2. Department of Computer Science, Shanghai Jiao Tong University, China
Email: {zhaoyiran0522, liutianyuan, zhangyang93}@sjtu.edu.cn

***Abstract*—In many countries, traffic signal information is unavailable to most drivers far away from the intersection. Thus, drivers might employ a relatively fast speed when they depart from the past intersection, while decelerate or undergo a complete halt at the intersection ahead. The often acceleration and deceleration driving pattern causes increased fuel consumption, air pollution, and even road accidents.**

**In this paper we devised and implemented a comprehensive speed-advisory driving system in which the central server utilizes the traffic signal information gained from random participation of smartphones to dynamically infer and calibrate the traffic signal schedule. As two basic functions, our system is able to determine best travel routes and advise optimal vehicle speed. Our system is also map-independent and infrastructure-less, meaning that it does not need any existing map database, nor does it depend on expensive devices or governmental assists. Our system does not eliminate breaks or stops at the intersection, but it will maximize the probability that drivers do not have to undergo a complete halt at intersections.**

**Results from campus road construction show that the construction of road-intersection topology is accurate; the timing schedule of individual intersection is inferred and calibrated fairly well. The average error of calculated road length is small, and the worst-case average timing error is about 2.4 seconds, meaning that the server can predict traffic signal schedule relatively well even if there is absent of acceleration signal at an intersection for a relatively long time. This is conducive to an accurate speed advisory system, which yields promising energy conserving results.**

***Index Terms*—Adaptive speed, Traffic signal schedule, Map construction, Smartphone sensor.**

## I. Introduction

Nowadays the traffic lights dominate city traffic, coordinating vehicles from different roads to safely transfer to other roads. But absent of traffic light information, drivers are hardly able to adopt appropriate speed and thus they often encounter total halt reaching the intersection. The complete stop means a complete loss of kinetic energy and the acceleration when the traffic light turns green would inevitably result in more fuel burn and air pollution.

Therefore, it is desired either to dynamically modify traffic signal phase timing to adapt to different traffic situations, or to inform drivers to adapt a proper speed, i.e., to deploy a Green Light Optimal Speed Advisory (GLOSA) system [1]. In the first case, great efforts have been made around the globe to establish Intelligent Transportation Systems (ITS). Typically, the SCATS system now is smoothing traffic flows in 154 cities in countries such as Australia, Singapore, China, New Zealand, etc., [1]. In particular, Singapore now possesses a state-of-the-art GLIDE system to gain information of traffic flow and calculate best traffic signal schedule. TrafficScan system in Singapore also helps to gather taxi speed information. But overall, most cities around the world still find this adaptive traffic signal system difficult to implement due to the requirement of large-scale infrastructure modification and complex specialized devices. In the second case, current approaches of GLOSA system have so far been based on roadside message signs that display the optimal speed drivers should maintain, or on the countdown timers at vehicular traffic signals that allow drivers to assume a proper speed [3]. Unfortunately, most of these systems are costly and impractical to deploy and maintain.

In this paper, we devise and implement a comprehensive infrastructure-less system to dynamically construct traffic signal information in a more realistic sense. As is shown in Figure 1, our proposed GreenDrive system only runs on smartphones and an Internet server, while depending on individual smartphones GPS and sensor data. The server of GreenDrive first mines GPS and smartphone sensor data to establish a road-intersection topological graph, and then gathers vehicle acceleration time and location to infer and formulate all traffic light phases at each intersection. When the server is ready to deduce future traffic signal schedule, it responds to individual drivers request and provides information about the intersection traffic state ahead. This process is based on the prior knowledge of travel routes, meaning that first the driver has to send a routing request to the server to calculate the best route. Then the server sends necessary road-intersection topology data and near-ahead intersections traffic light state information to the drivers smartphones, which then calculate the best speed and provide reasonable driving advice. Since our system does not eliminate complete stop at intersections, some vehicles would stop and wait at an intersection. But on detection of acceleration, the smartphone sends red-to-green transition signal to the server and the server thus could calibrate the timing of each phase and make future predictions more accurate. By real tests and careful modification of our system, we can achieve about a reduction of 90 percent of red light instances, saving about 20 percent of fuel on average.

Our GreenDrive system can provide the foundation of many other applications:

- *Commercial map revision and refinement.* Since our sys-

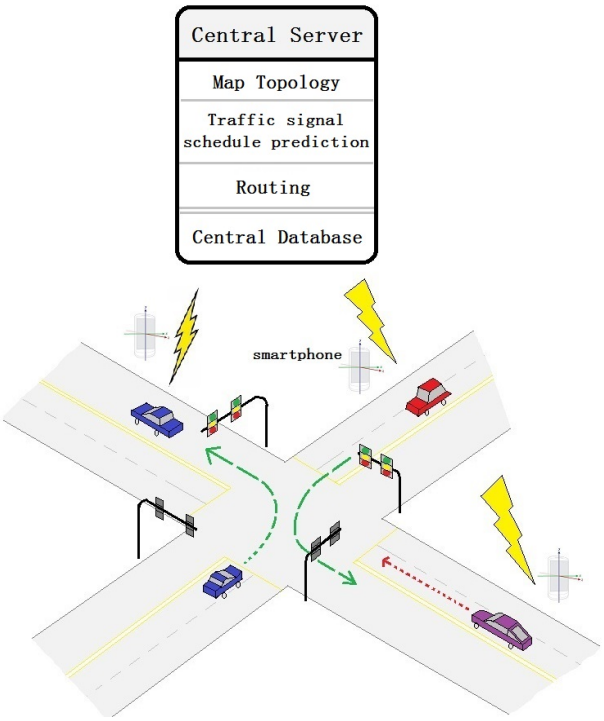



Fig. 1: GreenDrive System.

tem collects GPS information to infer road-intersection information, we can provide a considerable amount of trace data with fair accuracy. These data can be used to extract lane information and improve the accuracy of road geometry and intersection structure in commercial maps [4].

- *Traffic signal planning advisory service.* Currently, information about road congestion is gained from road cameras mounted on the traffic signal light, or from vehicle sensors in road pavement. However, not all intersections have such monitor devices due to their expensiveness. But our system central server does collect and provide average vehicle speed of a particular road. If the average speed is much lower than the system suggested speed, the system can infer road congestions and provide traffic signal adjusting suggestion to government agencies.
- *Driving behavior and road condition estimation.* Our system calculates and records vehicle acceleration in Earths coordinate system. And with combined information about vehicle travel direction, we can infer drivers behavior and suggests better commuting safety. In addition, the bumps and potholes can be detected by the smartphone sensors and thus proper road maintenance advice can be obtained. This functionality however, requires that the smartphone is mounted on a relatively fixed position with arbitrary angle to reduce false positives produced by user fiddling . But in our system, we do not implement this function and we simply shut down smartphone sensors to save battery power while in high speed, and activate them when the vehicle stops.
- *Red Light Violation Advisory (RLVA).* RLVA service warns the drivers when they try to speed up and squeeze through the intersection when the green light is about to turn. This is particularly dangerous when vehicles from perpendicular direction happens to encounter the transition to green light and maintains a relatively fast speed to sail across the intersection. When the smartphone detects acceleration while suggesting the driver to slow down because the time left is too short for the vehicle to pass through unless it exceeds the speed limit, warning will be given and driving safety record will be discredited.

The rest of this paper is organized as follows: In section 2 we give a brief analysis of related work. Then in section 3 we outline the system architecture. In section 4, we detail the methodologies and algorithms in the order of system phases, and give experimental results. In section 5 we evaluate the performance of our system and discuss other applications of our system as well as future improvements. Finally section 6 offers the conclusion.

## II. Related Work

There has been some infrastructure-less GLOSA systems that uses various sensors such as accelerometer and camera, together with advanced wireless communication technologies (ad-hoc, 3G, WIMAX, etc.), to infer traffic signal schedule.

Among such systems, SignalGuru [3] utilizes cell phone camera and adhoc network to collaboratively predict traffic light status ahead and advise drivers to maintain a reasonable speed. However, this system is hard to implement in a real sense because it may fail to gather traffic light information when there are not enough vehicles on road since it uses opportunistic ad-hoc communications to collaboratively learn the timing patterns. Also, traffic signal information gathered by smartphone cameras mounted on windshield may also be problematic when the traffic light is deformed or occluded, or when it is raining and camera view is obstructed. Furthermore, SignalGuru requires carefully positioned camera angle to pick the correct traffic light when facing a complex intersection.

In view of a more reliable means of communication and a more robust system, some other methods are proposed. Guobao Ning et al. [5] designed an Adaptive Driving Speed Guiding system (ADSG) that uses Internet server to gather location information from cell phones and calculate optimal speed from traffic light timing downloaded beforehand. However, this ADSG system is not experimented or implemented as it depends on pre-downloaded traffic signal information from the database owned by the ministry of transportation, which may be inaccurate.

To make the system more applicable, our proposed system automatically extracts road-intersection topological information and traffic signal timing, while effectively maximizes the probability that individual vehicle does not encounter a red traffic light. Our GreenDrive system differs from other smartphone-based systems in that our system is independent of additional information from either third-party map database or from governmental agencies, and is able to deal with relatively complex intersections with multiple sets of traffic lights and

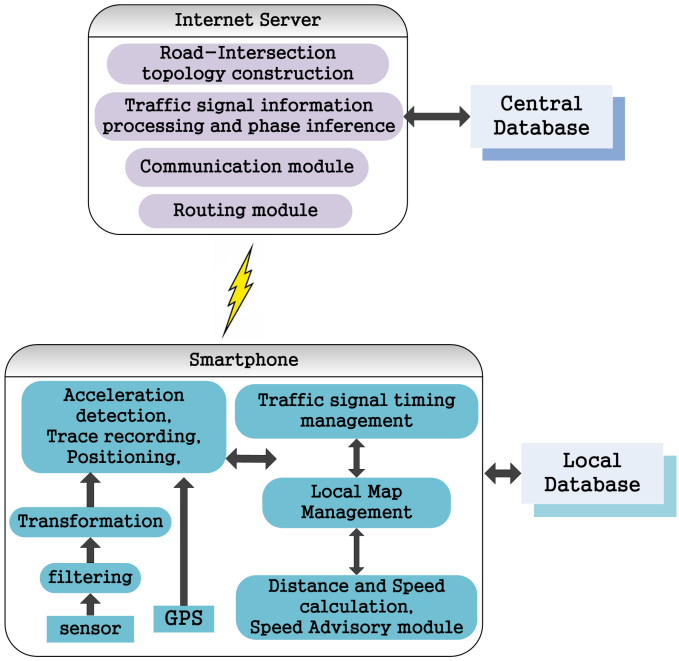



Fig. 2: GreenDrive System Modules.

various phase sequence patterns. Our system also requires that drivers input destination so that travel routes can be determined by the server and that the targeted traffic signal information is specified. In addition, the proposed system does not require careful position of smartphones, and every passenger could use their own smartphone as the driving guider and hold it in hands since our system does tolerate normal user fiddling. In a word, this proposed system is more likely to be implemented in real world.

## III. System Architecture and Procedure Description

The general system modules are illustrated in Figure 2. We exploit smartphone capabilities and set up a computer as the server to implement the whole system.

### A. Android smartphone capabilities exploited in our system:

1) *3-axis Accelerometer.*
The on-device accelerometer measures the acceleration applied to the device in its own coordinate system, including the force of gravity. Apart from acceleration sensor in android phones, the android OS 2.4 and later also provide gravity sensor interface which estimates the direction of gravity using the same accelerometer in its own coordinate system. The gravity information helps us to calculate the acceleration direction and magnitude in real earths coordinate system. Although the gravity measured is not strictly the real gravity (instead it is aligned with acceleration direction but has a magnitude of earths gravity), it does not significantly affect the transformation from phones coordinate system to earths coordinate system.

2) *3-axis Magnetometer.*
Smartphones also integrates magnetometer on device to measure the geomagnetic field in its own coordinate system. The reason of using the magnetometer is that we have to utilize the measurements of geomagnetic field as a reference to transform the acceleration vector from the devices coordinate system to earths coordinate system. Although the readings of magnetometer contains a lot of noise and is sensitive to indoor man-made magnetic field, it is generally assumed that there is no strong magnetic field in vehicular environment, and the noise is going to be reduced to an acceptable level by simple filtering.

3) *Global Positioning System (GPS).*
GPS accuracy is greatly improved following the advent of DGPS system. Smartphones used in our test show that the average accuracy of position is about 4-6 meters, with a minimum of 3 meters when the conditions fit. The GPS system provides smartphones with information about devices position (longitude and latitude), speed (in meters per second), bearing (heading direction, in degrees), UTC time (in milliseconds since January 1, 1970), accuracy (in meters), etc. Given the sufficient accuracy of GPS information, we depend to a large degree, on smartphones GPS data.

### B. Road-intersection topology construction

The initialization of the system is to construct road-intersection topology. Volunteers using this application in vehicles are needed to record information gathered by on-device sensors and GPS, and then transmit acceleration data and GPS traces to the central server when wifi is available. To locate intersections, we group into clusters the acceleration (from a halt that lasts at least 20 seconds) vectors that are often the densest at intersections with traffic lights. The reason to use this method is that two crossing GPS traces do not necessarily indicate the existence of intersections with traffic lights. To improve the robustness of intersection identification, the pattern of acceleration vectors within a cluster is analyzed so that accelerating on road segments upon emergency or congestion can be recognized and filtered away. The identification of intersections will be done only once unless there is need for update. Volunteers at this initialization phase should try to avoid using this application in parking lots or residential areas so as not to incur false positives. And even if false intersection is produced, it does not affect the systems future functioning. The resulting intersection should contain the direction and number of each branch.

After sufficient number of intersections is identified (this number should be in accordance with estimated intersection density and the area of the region), the server now uses recorded GPS traces from smartphones to link the intersections and establish a topological graph with minimum database size. The identification of intersections and the generation of road segments can be concurrent in later stages as new information is gathered from future participants. Now drivers could download the map data of the vehicles neighboring region and store it in local database.

### C. Traffic signal schedule inference and update

Based on the existing road-intersection topology, future stop and acceleration information will be associated with a corresponding intersection that has a unique ID. In this stage, the traffic signal schedule should be re-established in the early morning following the interval of night time. And some drivers will be contributors since they transmit intersection phase transition information to help server establish timing schedule while no immediate benefit is returned. The central server first receives information from accelerated vehicles that contain the intersection ID, the branch on which it has waited, the branch on which it departs from the intersection, and the time interval from acceleration to transmission. Given the shape and pattern of individual intersection, the traffic signal cycle length is obtained, and the sequence and flow pattern of each phase is inferred. Later on, the cumulative prediction error would cause some vehicles to stop and accelerate again. But such acceleration information is used by the server to calibrate its prediction. The calibration process minimizes the mean square error between servers prediction and real event time. Each calibration would yield the length of each phase and a reference starting time. We assume that the cycle length and phase length remain relatively stable, but adjustment of traffic signal schedule is acceptable.

### D. Routing service and speed advising

To suggest proper speed to the next intersection, the smartphone has to gain prior knowledge of travel route. When the drivers launch this system, a destination has to be specified. Then the drivers start location and destination location is sent to the server to calculate the best route. The server chooses the route with least travel time by dividing the length of each road segment by the average speed recorded. If the current average speed on a certain road is not available, the server assumes that all drivers travel on that road with speed up to the legal limitation. In other words, the server chooses a route that is least in distance when information about road traffic condition is absent. And once the route is determined and downloaded to the smartphone, the smartphone requests traffic signal schedule information of the next two intersections. And then with help of local database and the vehicles current location and heading direction, the distance to the next intersection is calculated and the optimal speed is advised. Together with future requests for traffic signal information, the smartphone also sends the average speed of the last road segment so that the server can provide other vehicles with better real-time routing.

## IV. System implementation and algorithm

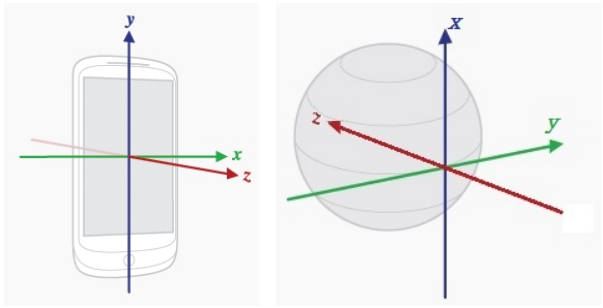
### A. Acceleration module

This module is important in that the system depends heavily on acceleration data in various stages. Since the measurements from smartphone 3-axis accelerometer contain a lot of noise, the acceleration data is first filtered with a cut off frequency of 2Hz. To further increase detection accuracy, we project the acceleration vector onto the direction which the vehicle heads just before a complete stop. The vehicles direction is gained from GPS information and is maintained by the system once the vehicles speed is lower than 1 meter per second.

*1) Definition of the Body Coordinate System and the Local North-East-Down(NED) Coordinate System:*

The smartphones body coordinate system is illustrated in Figure 3(a). The X axis is horizontal and points to the right, the Y axis is vertical and points up and the Z axis points towards the outside of the front face of the screen. The acceleration measured by smartphones is given in this body coordinate system.

As is shown in Figure 3(b), in Local NED coordinate system, the X axis is pointing to the North Pole, and is tangential to earths surface. The Y axis is tangential to the ground at the device's current location and points towards the east. And the Z axis is pointing to the sky, which is vertical to both X and Y axes. The GPS bearing information is provided as the angle between heading direction and the X-axis in the Local NED coordinate system.




(a) Body Coordinate System (b) Local NED Coordinate System

Fig. 3: Two coordinate systems used in our system.

*2) Coordinate system transformation:*

To detect acceleration along the vehicles travelling direction, the acceleration vector gained from smartphones body coordinate system has to be transformed into the Local NED coordinate system. The transformation follows Z-Y-X rotation sequence. Denote $\theta_z,\ \theta_y,\ \theta_x$ as the angle of rotation about intermediate Z, Y, X-axis in sequence. Thus the rotation matrix $\boldsymbol{R_{nv|b}}$ from the body frame to the Local NED frame and is given by [6]:

$$\boldsymbol{R_{nv|b}} = \begin{bmatrix} C(\theta_y)C(\theta_z) & C(\theta_y)S(\theta_z) & -S(\theta_y) \\ \begin{matrix} S(\theta_x)S(\theta_y)C(\theta_z) \\ -C(\theta_x)S(\theta_z) \end{matrix} & \begin{matrix} S(\theta_x)S(\theta_y)S(\theta_z) \\ +C(\theta_x)C(\theta_z) \end{matrix} & S(\theta_x)C(\theta_y) \\ \begin{matrix} C(\theta_x)S(\theta_y)C(\theta_z) \\ +S(\theta_x)S(\theta_z) \end{matrix} & \begin{matrix} C(\theta_x)S(\theta_y)S(\theta_z) \\ -S(\theta_x)C(\theta_z) \end{matrix} & C(\theta_x)C(\theta_y) \end{bmatrix}$$

Where $C(\theta),\ S(\theta)$ denote $\cos(\theta),\ \sin(\theta)$, respectively.

To calculate the rotation matrix, we have to use the two reference vectors, i.e. the gravity and the geomagnetic field. Strictly speaking, using the direction of geomagnetic field as the X-axis in Local NED frame is not accurate. Since the magnetic north is different from geodetic north. But in our system, that difference can be ignored. Thus, the acceleration vector $\vec{\boldsymbol{a}}_{\boldsymbol{nv}}$ in Local NED coordinate system can be expressed

by:

$$\vec{\boldsymbol{a}}_{\boldsymbol{nv}} = \begin{pmatrix} a_{x_{nv}} \\ a_{y_{nv}} \\ a_{z_{nv}} \end{pmatrix} = \boldsymbol{R}_{\boldsymbol{nv|b}} \begin{pmatrix} a_{x_b} \\ a_{y_b} \\ a_{z_b} \end{pmatrix} = \boldsymbol{R}_{\boldsymbol{nv|b}} \cdot \vec{\boldsymbol{a}}_{\boldsymbol{b}}$$

Where $\vec{\boldsymbol{a}}_{\boldsymbol{b}}$ is the acceleration vector in body coordinate system.

*3) Vehicle acceleration detection:*

The calculated acceleration vector in the Local NED coordinate system is then projected onto the vehicles heading direction just before its halt. This is based on the assumption that usually a vehicle that decelerates and stops at an intersection remains a relatively stable direction. This direction is fairly accurate from GPS information, and the low speed threshold to hold and record the last heading direction is empirically set to 1 meter per second. The projection naturally filters away noise or user fiddling acceleration perpendicular to the vehicles heading direction. And to further reduce false positive produced by user fiddling, we set up a time window to calculate the aggregated acceleration in duration of about one second. In the case of user fiddling, such as screen touching, moving or even shaking, the movement is usually less than one second so that the aggregated acceleration is neutralized by deceleration of approximately the same magnitude.

The filtering, transformation of coordinate system and taking aggregated acceleration allow for arbitrary positioning of the smartphone and user fiddling. Real testing reveals that this method is robust and is able to detect 99 percent of vehicle acceleration from zero speed.

### *B. Construction of road-intersection topology*

Since our application requires little volunteer efforts, we can have a large number of participants in a variety of city regions. Once wifi is available, they will send recorded acceleration and GPS data to the central server.

*1) Identifying intersections:*

The server collects all acceleration location and direction information to find out the position of intersections. We assume that most volunteers use this system only when in real road environment instead of in parking lots or residential areas. Thus a cluster of acceleration location vectors within a circle of approximately the size of an intersection resembles a real intersection in high probability. So we use a method similar to mean shift to locate dense acceleration points. Through successive computations of the mean shift, the center of the circle moves along a path leading to a local vector density maximum. Such process is illustrated in Figure 4.

Then the vectors in the final cluster circles should be further analyzed to rule out noise occurred on the road segments or in parking lots. Within those final circles, we extract direction data and use k-means algorithm to group points by their heading direction. If the number of groups is less than three, we discard this circle because it might be on a straight road segment. If the number of direction groups is larger than five, we also dont believe that it is a real intersection. Then we find the center of each group that and check if the average direction vectors in each group are pointing into the polygon formed by those group centers. This process is shown in Figure 5. Only

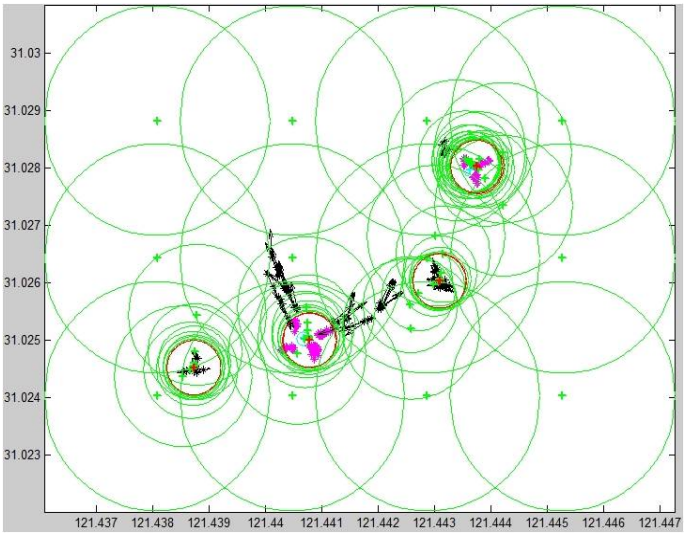
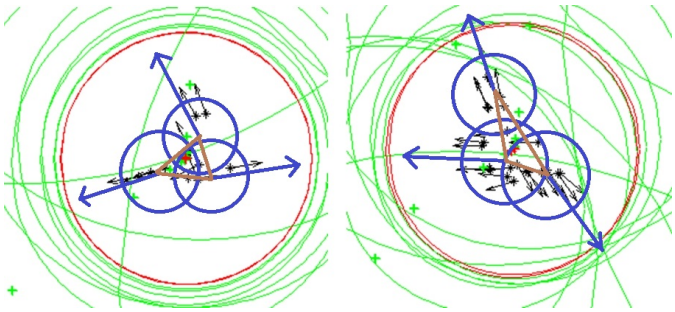
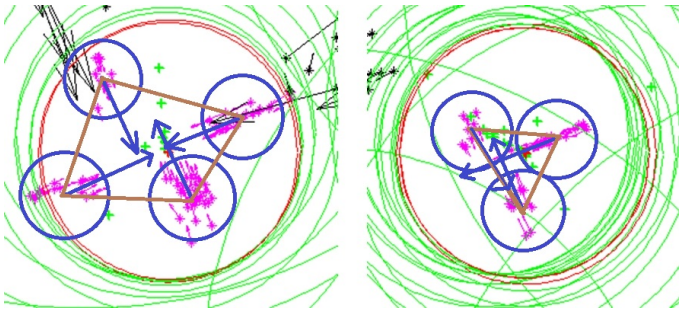



Fig. 4: Mean Shift to locate intersection position.

valid group pattern form a candidate intersection. Then we give each intersection a unique ID and number each branch to establish an embryonic database.



(a) Bad Clusters



(b) Good Clusters

Fig. 5: Filter out bad cluster circles.

*2) Linking road intersections with GPS traces:*

Based on the embryonic intersection database, we give each GPS trace an opportunity to form a road segment starting from one intersection to another. From a valid group of acceleration points that share approximately the same direction, we select each at a time one vehicle ID and backtrack to the previous intersection along its trace. To fight against GPS data inaccuracy, we deploy an algorithm proceeds in a fashion similar to weighted mean shift algorithm that finds the centroid and mean direction of a cluster of GPS points on the same side

of the road for every 10 meters. The weight of each GPS point is corresponding to the accuracy of its position, and the resulted circle is about the size of a typical road width. We name the center of the final circles anchor point (with direction). Although we deploy mean shift every 10 meters, only a change of mean direction that is larger than 20 degrees or when the distance to the last anchor point is larger than 100 meters will resolve to a valid new anchor point (Figure 6). After each vehicles trace is processed (if another vehicles trace lies in the processed segment, it will jump to where the trace departs into a different branch), all roads linking to the source intersections branch is recorded as groups of anchor points with direction.

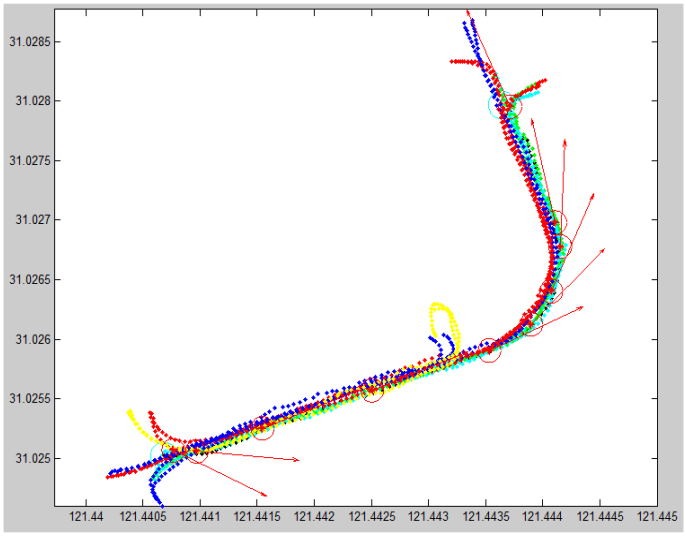


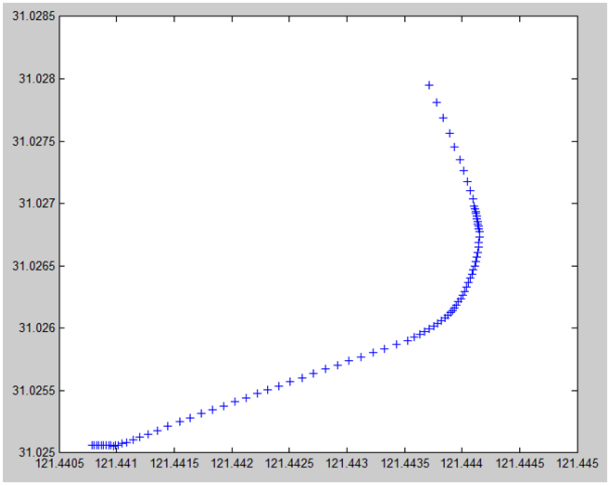

Fig. 6: Generation of anchor points of a road segment.

Then we use modified second order B-splines to represent road lines as piecewise defined polynomials with first order continuity at the anchor points. Note that the B-spline control points are not anchor points. Instead, each time three control points are calculated given two anchor points to formulate the second order polynomial that passes through the two anchor points. The result if shown in Figure 7. After all roads between any pair of intersections are represented as piecewise polynomials, the distance of each road segment is calculated; hence the complete topological graph of a region is established.

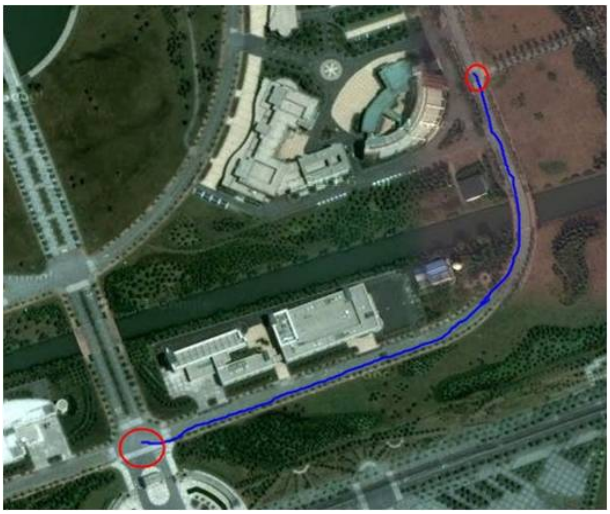
*3) Constructed central database:*

The intersection ID, its position, the numbered branches in this intersection, the distance and each anchor points information is stored in the servers central database, as is illustrated in Table I.

Upon each request from smartphones for map data download, the server collects the necessary road-intersection information along the vehicles route and sends the package to individual smartphones. When speed information is available, it will be added to the corresponding road segment so that server could provide better routing service.




(a) Result from B-spline representation



(b) Compare to real road in Google Map

Fig. 7: Results from construction of road segment.

## C. Traffic signal schedule inference and update

Since each intersections shape and linking topology are stored in the central database, the server is able to dynamically infer and calibrate traffic signal phases and timing schedule. In the following statements, we take into consideration the fact that there might be a long queue of vehicles waiting at an intersection. So as the traffic light turns green, the first vehicle's acceleration time is significantly earlier than the last one. To deal with this situation, the server only takes into account the first transition signal of each phase, subsequent accelerating vehicles' messages will be discarded if their event time is following closely to the first event time of the same phase.

*1) Traffic signal phases and phase sequence inference.:*

This is the initial phase of establishing traffic signal schedule. It takes far less time to complete than the construction of the road-intersection topology. The traffic signal schedule is probably inaccurate or even erroneous without calibration from enough traffic flow during night time. As the number of vehicles increases in early morning, our system is able to

TABLE I: Constructed database.

| Intersection ID | longitude | latitude | $R_n$ direction | $R_n$ connected intersection | length of $R_n$ | A.P. number | A.P. longitude | A.P. latitude | A.P. direction |
|---|---|---|---|---|---|---|---|---|---|
| 001 | 116.3463 | 29.1354 | 15 | 00n | 210 | 10 | 121.453 | 30.323 | 353 |

re-establish and keep track of the traffic signal schedule. To discard erroneous timing data, every time the system predicted traffic phase timing disagrees with phase transition signal sent by smartphones to a certain level, the system will wipe out history traffic signal schedule and restart. Whatever the shape and pattern of an intersection is, we assume that the traffic signal cycle length, the length of each phase, and the sequence of the phases are the same within a relatively long period. If scheduling is changed and the predicted timing shows significant error, our system will restart this stage.

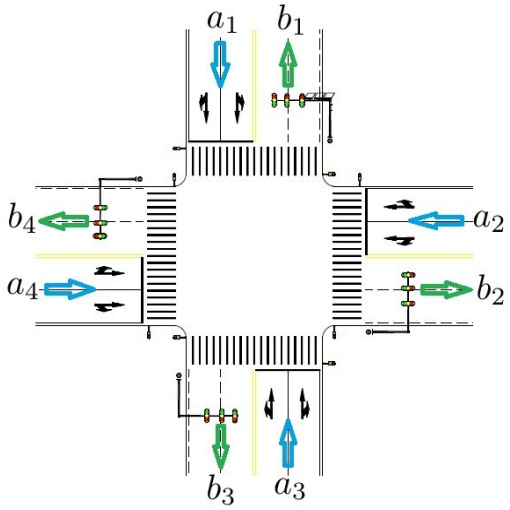
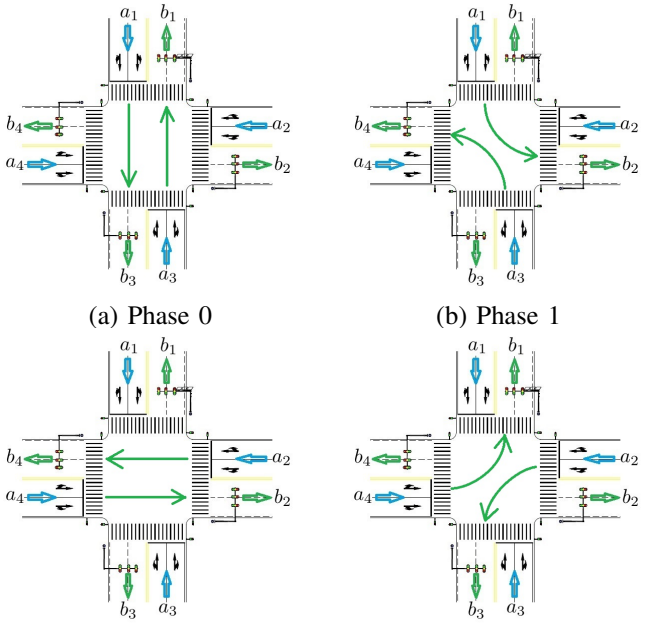



Fig. 8: A typical road intersection with $n = 4$.

To clarify branch names for later use, here the in-coming branches of an intersection are denoted by $\{a_i, i = 1, 2...n\}$ (n is the number of branches), and the out-going branches are labeled $\{b_i, i = 1, 2...n\}$. Typically, $n$ equals to 3 or 4. Taking $n = 4$ as an example in Figure 8. Here we denote $(a_i, b_j)$ as the traffic flow from $a_i$ to $b_j$. Right turns and turn back such as $(a_i, b_i), i = 1, 2, 3, 4$, and $(a_2, b_1)$, $(a_3, b_2)$, $(a_4, b_3)$, $(a_1, b_4)$ are not taken into account in our system. If any two types of traffic flow happens at the same time, say, $(a_1, b_2)$, $(a_3, b_4)$, then they form a phase.

To infer the number of phases and their sequence, it is required that each smartphone transmit its in-coming branch ID, its out-coming branch ID and the interval of time between acceleration occurs and transmission occurs to the central server. This is possible since each smartphone has the topology map of its visiting region. It is further assumed that in each traffic signal cycle, every phase of traffic flow $(a_i, b_j)$ only happens once, and that drivers obey traffic rules so as not to run the red light. Based on such assumptions, it is reasonable that acceleration at intersections from a halt which lasts at least 20 seconds is the signal of a transition from red to green of the corresponding phase. So during this initialization, the server should have collected sufficient amount of data of each intersection so that many pairs of flow $(a_i, b_j)$ are grouped into phases. If the phase number is reduced to a reasonable one, say, four phases for a $n = 4$ intersection (typically like Figure 9), then the server goes into finding the length of traffic signal cycle $T$.




(a) Phase 0 (b) Phase 1

(c) Phase 2 (d) Phase 3

Fig. 9: Caption. Typical phases in an intersection with $n = 4$.

Using the same collected data, the server finds the minimum time interval of any repeating phase $S_i$, $i = 0, 1, 2, 3$. With high probability, that minimum time interval (it also should be reasonably large) is approximately the length of the traffic signal cycle $T$ (although it may be smaller due to deviation). Given the rough cycle length $T$, the server then tries to figure out the sequence of the phases. Again, using the same data (perhaps more are collected), the server finds the closest next phase, say $S_1$, $S_2$, $S_3$, $S_0$, following each phase $S_0$, $S_1$, $S_2$, $S_3$, respectively. Denote the time interval $dts_i = t(S_{next}) - t(S_i)$ as the time from the start of phase $S_i$ to the start of its closest next phase. If $\sum_{i=1}^{n} dts_i$, $(n = 4)$ approximately equals to $T$ (with difference sufficiently smaller than $T$ itself), then the server deems the initialization process successful.

*2) Traffic signal timing calibration and update.:*

Once the number of phases, the sequence of the phases, and the approximated length of phases are determined, the server takes one phases starting time as the servers prediction begin time. Denote $t_s$ as the prediction starting time, and $ts_k$ as the server-received start time of phase $S_k$. Upon receiving

| $t_s$ | | | | |
|---|---|---|---|---|
| $ts_k$ | $ns_0$ | $ns_1$ | $ns_2$ | $ns_3$ |
| $ts_l$ | $ns'_0$ | $ns'_1$ | $ns'_2$ | $ns'_3$ |
| $ts_p$ | $ns''_0$ | $ns''_1$ | $ns''_2$ | $ns''_3$ |
| ... | ... | ... | ... | ... |

TABLE II: Set of data for calibration.

$ts_k$ from smartphones, the server judges the number of times each phase has happened. To be more specific, using $t_s$, $ts_k$, $T$ and $dts_k$, the server will infer whether the prediction is ahead of real timing or is lagging behind. Denote $ns_i$ as the number of times phase $S_i$ $(i = 0, 1, 2, 3)$ has happened from $t_s$ to $ts_k$. The server calculates $ns_i$ and records the set of data $ts_k$, $ns_i$ $(i = 0, 1, 2, 3)$ into an array (illustrated in Table II) for calibration.

As the received signal continues to add new rows in the Table II, we dynamically make a calibration when the size of rows reaches a threshold ($i.e.$ $N$, this value also changes according to the level of prediction error). Resolve the Table II into five columns (without the first row), each forming a vector, namely, $\boldsymbol{ts}, \boldsymbol{ns_0}, \boldsymbol{ns_1}, \boldsymbol{ns_2}, \boldsymbol{ns_3}$. With the five vectors, we are going to find the calibrated starting time ($t_0$) and calibrated phase length ($dts_i$, $i$= 0, 1, 2, 3).

The goal is to find a best estimation of $t_0$, $dts_i$, ($i$=0, 1, 2, 3) that have the minimum mean square error (MSE) with the real event time. So we formulate a target function that represents the MSE:

$$\begin{aligned} &F(t_0, dts_0, dts_1, dts_2, dts_3) = (t_0 - t_s)^2 \\ &+ \textstyle\sum_{i=1}^{N}(t_0 + \boldsymbol{ns_0}(i) \cdot dts_0 + \boldsymbol{ns_1}(i) \cdot dts_1 \\ &+ \boldsymbol{ns_2}(i) \cdot dts_2 + \boldsymbol{ns_3}(i) \cdot dts_3 - \boldsymbol{ts}(i))^2 \end{aligned} \tag{1}$$

Where $N$ is the size of the vectors $\boldsymbol{ns_i}$, $\boldsymbol{ts}$. Minimize this target function by taking partial derivative of each variable in Equ.(1):

$$\frac{\partial F}{\partial t_0} = 0, \ \frac{\partial F}{\partial dts_i} = 0, \ i = 0, 1, 2, 3$$

Solving the resulting equations yields calibrated $t_0$, $dts_i$, $i$= 0, 1, 2, 3. Note that if in rare cases the function cannot be solved, or that the matrix is singular, we cannot get the individual value of every $dts_i$, but the sum of some of them, and then allocate value based on their history ratio. In most of our tests, the MSE algorithm utilizes a fair amount of history data to provide sound calibration to future predictions.

### D. Server and smartphone communication

There are various messages sent between the smartphones and the server. As is mentioned, the smartphone should transmit its acceleration and GPS data to the server. In addition, the smartphone sends requests for routing and traffic schedule information. It also uploads the average speed information and possibly, road condition information to the server. Apart from doing calculations, the server has to deliver the best travel route to smartphones, and sends traffic signal schedule information for individual smartphone.

*1) Routing:*

When a driver launches our application, the destination is specified and sent to the server for request of routing. The server extracts information of map topology and average speed within the circle region (a circle that centers at the middle of start point and the destination point, with diameter slightly larger than the distance between the two points), and employs Dijkstra algorithm to calculate the route with the least travel time.

*2) Best speed calculation:*

As the vehicle advances, it will request for traffic signal schedule of the two intersections ahead on its route. And the distance from the vehicles current position to the intersection is calculated in a similar way as 4.2.2. The length of the B-spline between anchor points to the intersection is calculated and the distance of the vehicles position to the nearest anchor point is also obtained through calculation every 10 seconds. Combined with the distance and the traffic signal schedule, the optimal speed should be such that (in Figure 10) the speed line should remain as straight as possible on the two road segments.

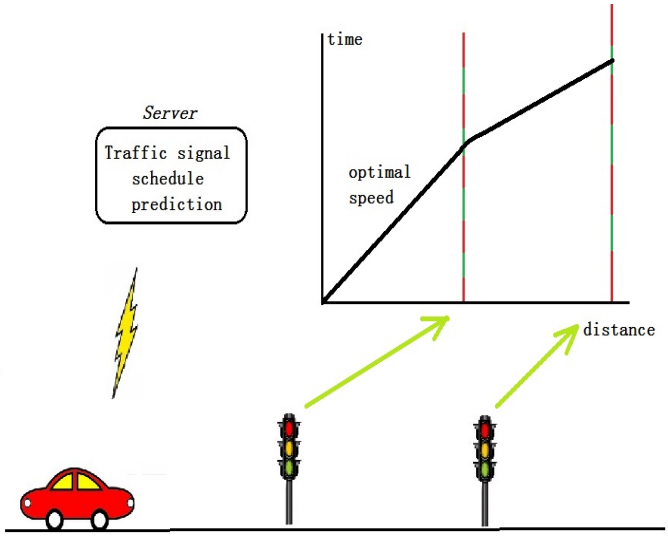



Fig. 10: Obtaining optimal speed.

*3) Traffic and road condition upload:*

The server uses average speed of road segment to calculate the best route, to infer road congestion, and to provide suggestion for traffic signal planning. If the actual speed of vehicles is significantly below the advised speed, the smartphone will send a signal of congestion along with the average speed. And if the number of congestion signal is too large in ratio with the roads capacity, the server will propose a traffic signal re-planning. In addition, if the accelerometer also detects large measurements along the Z-axis of the coordinate system, it will send a signal of bad road condition.

*4) Special cases:*

It is often the case where the driver wants to go on his or her frequent-traveled route instead of that suggested by the server. Our system does allow deviation from preset route. Upon detection of turning into another branch, the smartphone will send a message for re-routing. And procedure of finding

a route is operated again, and new intersection traffic signal schedule is sent to the smartphone.

In some cases, the driver just does not want to drive as fast as the system suggests. This is acceptable because besides the optimal speed (the fastest one within legal limit), a lower speed that allows the driver to sail across the intersection in the next cycle is also proposed on the interface.

## V. Performance evaluation

Real tests are yet to be carried out. Those tests include:

1) Acceleration detection test. We should get in a real vehicle and test the acceleration module, to gain the statistic result of the detection rate.
2) Campus road-intersection topology construction. We need to collect acceleration and GPS data using real vehicles and take normal trips in SJTU. Or we can find some real intersection-dense areas to see if the topology graph can be established.
3) Real intersection signal schedule prediction. We need to set up the server and call on several volunteers to simulate acceleration in each branch of a targeted intersection. To test the overall performance of traffic signal schedule prediction would be unrealistic and may be replaced by complex simulation.

## VI. Conclusion

We believe that when the tests are carried out, the energy consumption shall be reduced, and information about road condition and traffic signal re-planning should be available.

## Acknowledgment

The author would like to thank Prof. Xinbing Wang, Xiaohua Tian, Tuo Yu, etc. who offer us great help.

## References

[1] J. van Leersum, “Implementation of an advisory speed algorithm in transyt,” *Transportation Research Part A: General,* 1985.

[2] Wikipedia: Sydney Coordinated Adaptive Traffic System, `http://en.wikipedia.org/wiki/Sydney_Coordinated_Adaptive_Traffic_System`

[3] Emmanouil Koukoumidis, Li-Shiuan Peh and Margaret Mar tonosi, ”SignalGuru: Leveraging Mobile Phones for Collaborative Traffic Signal Schedule Advisory,” in *Mobisys,* 2011.

[4] Stephan Schroedl, Kiri Wagstaff, Seth Rogers, Pat Langley, and Christopher Wilson, “Mining GPS Traces for Map Refinement,” *Data Mining and Knowledge Discovery,* 9, 59C87, 2004.

[5] Guobao Ning, and Lushen Cai, “Adaptive Driving Speed Guiding to Avoid Red Traffic Lights,” in *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering,* 2013.

[6] G.Cai, “Unmanned Rotorcraft Systems, Advances in Industrial Control” *Springer-Verlag London Limited,* DOI $10.1007/978-0-85729-635-1-2$, 2011.

**Yiran Zhao** is currently pursuing his B.E. degree in electronic engineering in Shanghai Jiao Tong University, China. His research interests are in electronic systems.

**Tianyuan Liu** is currently pursuing his B.E. degree in electronic engineering in Shanghai Jiao Tong University, China. His research interests are in computer science.

**Yang Zhang** is currently pursuing his B.E. degree in electronic engineering in Shanghai Jiao Tong University, China. His research interests are in computer science.