Unsupervised Annotated City Traffic Map Generation

Rohit Verma, Surjya Ghosh, Aviral Shrivastava, Niloy Ganguly,
Bivas Mitra, Sandip Chakraborty
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur, INDIA 721302
{rohit.verma,surjya.ghosh}@iitkgp.ac.in, aviralshrivastava93@gmail.com,
{niloy,bivas,sandipc}@cse.iitkgp.ernet.in

ABSTRACT

Public bus services in many cities in countries like India are controlled by private owners, hence, building up a database for all the bus routes is non-trivial. In this paper, we leverage smart-phone based sensing to *crowdsource* and populate the information repository for bus routes in a city. We have developed an intelligent data logging module for smart-phones and a server side processing mechanism to extract roads and bus routes information. From a 3 month long study involving more than 30 volunteers in 3 different cities in India, we found that the developed system, *CrowdMap*, can annotate bus routes with a mean error of $10m$, while consuming 80% less energy compared to a continuous GPS based system.

CCS Concepts

•Human-centered computing → Ubiquitous and mobile computing systems and tools;

Keywords

Crowdsourcing; Map generation; City transports

1. INTRODUCTION

According to IBM's Commuter Pain Survey [1] conducted in 2011 over $8,042$ commuters in 20 cities on six continents, 41% of the respondents believed that improved public transport information would help in better planning, and 25% would highly appreciate efficient information of road conditions. Essentially, real time commute information may facilitate the commuter in route selection, on the fly, based on the comfort level and travel time. Notably, the immediate best solution of Google Transit are mostly unavailable in cities of developing countries.

Constructing any commute facilitator-cum-recommender system relies on building a proper information repository. In developing countries, populating this information repository is a major challenge. Albeit there may be several possibilities such as manually annotating the Google map or relying on the city transport authorities. However, almost all commuters travel with smart-phones which can seamlessly sense the road and traffic condition, and such sensing is also free from direct engagement of the owner. In this paper, we leverage on such user-engagement free technology to *crowdsource* and populate the information repository.

Nevertheless, multiple challenges need to be tackled to effectively implement such mobile sensing approach. First, to ensure least user participation and energy usage, the system should intelligently decide when to start and stop data logging. Second, automatically identifying and tagging landmark locations in an energy efficient way. Third, generate the travel trajectory of the user in presence of minimal GPS information.

Studies like [4] have explored the capabilities of various onboard inertial sensors, like accelerometer, gyroscope, compass, to monitor unique road signatures or landmarks. There are also researches on inertial navigation that uses sensors to estimate and track the position of moving devices [3]. Further, some studies like [2] have proposed techniques for producing route maps from smart-phone data or opportunistically collected GPS traces. They primarily rely on war-driving data, with significantly less possibility of noise and are also limited to particular scenarios of smoothed data (like taxi movement in a city of a developed country).

In this paper, we propose a *crowdsource* based solution, *CrowdMap*, which seamlessly collects travel data and generates the trajectory followed by the user. The system has two modules – *data collection* that runs at smart-phones, and *trajectory generation* that processes the data at *CrowdMap* server to generate the travel trajectory with bus routes related information. We have conducted a 3 month long study involving more than 30 volunteers in 11 different routes of 3 different cities in India (§3). We observe that *CrowdMap* can detect the landmarks with an average accuracy of 95% and annotate the route on real map with an error of 5 meters at worst case.

2. THE CROWDMAP SYSTEM

Broadly, *CrowdMap* has two major modules, as shown in Figure 1; (a) the data collection module that runs on smartphones, (b) the data processing module that runs on the *CrowdMap* server.

2.1 Data Collection on Smart-Phones

The crowdsourcing based data collection module is designed to ensure intelligent logging of data only when the user is in a bus with minimum user engagement and detect

SIGSPATIAL'16, October 31-November 03, 2016, Burlingame, CA, USA

DOI: http://dx.doi.org/10.1145/2996913.2996942

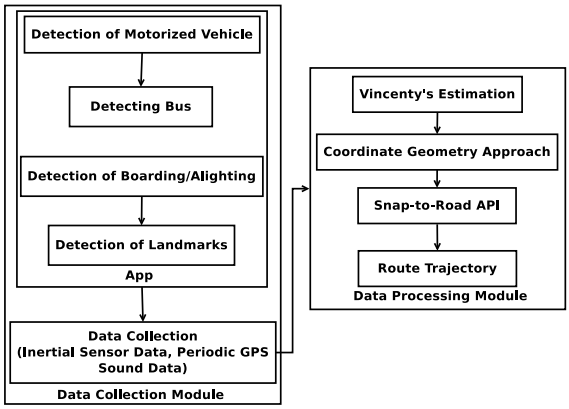

Figure 1: CrowdMap: System architecture

location of specific route signatures, called landmarks (like speed breakers, turns) using smart-phone sensors, with minimum strain on the smart-phone battery. Finally, the data logger transfers the accelerometer and the compass readings for the duration of bus commute, and the sequence of GPS annotated landmarks encountered on the route to the server.

2.1.1 *Smart Data Logging*

In *CrowdMap*, we log the smart-phone sensor data only when the commuter travels on a bus. This smart data logging activity gets accomplished by first recognizing that the commuter is traveling in a motorized vehicle from the initial accelerometer trace; accordingly this module initiates the smart-phone sensor sampling process. In the next step, the second level the module leverages on the sound sensor data for the first few minutes to classify whether the motorized vehicle is a bus. Once the module detects that the user is traveling by a bus, it continues collecting the inertial sensor data. The detail methodology follows.

Detection of Motorized Vehicles: The key idea is, the average acceleration in y-axis (in the direction of movement) for a motorized vehicle (like cars, bus etc) is distinctly higher than the other modes such as stationary, walking, bicycle, etc. Figure 2(a) exhibits a striking difference between the motorized vehicles and non-motorized modes. It is clearly evident that acceleration in Y-axis is significantly higher for both the motorized vehicles (bus and car) when compared to, say bicycle. Notably, Figure 2(a) shows that even in case of congestion, this smoothened feature can discriminate the motorized vehicles (*"bus congested"* in the plot). We also see that it is not possible to classify the different motorized vehicles this way.

Next issue is to compute the latency in classifying the motorized commuters. We compute the average acceleration over the first k minutes on the smoothened data to filter out non-motorized commuters. Figure 2(b) confirms the fact that considering $k = 2$, we obtain a decent contrast between motorized and non-motorized commuters.

Extracting Bus Commuters: In the next step, we perform further stratification to filter the commuters traveling on bus, the duration only for which we collect data. In order to identify buses, we leverage on the smart-phone audio sensor and extract features to distinguish bus commuters from other motorized vehicles.

As a proof of concept, we collect sound sensor data for

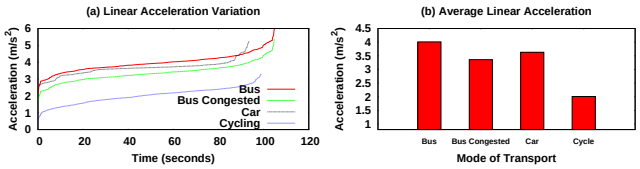

Figure 2: Acceleration variation for motorized vehicles versus cycling; (a) signature of the acceleration along y-axis for different modes (b) Average value of the acceleration along y-axis calculated over the first 2 minutes

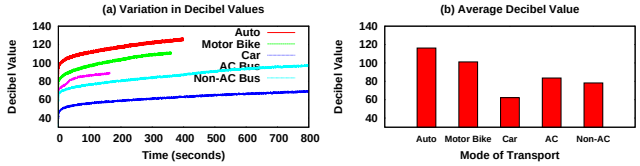

Figure 3: Variation in sound data for different motorized vehicles; (a) Signature of the sound data for different modes (b) Average value of the sound data calculated over the first 4 minutes

different motorized (around 60) vehicles. Clear discrimination in decibel scale between bus and other vehicles can be observed in Figure 3(a). Moreover, Figure 3(b) exhibits the fact that in ($k = 4$) minutes, we can filter the bus commuters from the crowd.

2.1.2 *Detection of GPS Annotated Bus Stops and Landmarks*

In order to initiate the data logging activities, one needs to identify the location of bus stops and commuter's boarding & alighting on a bus. Additionally, detection of landmarks, along with their GPS locations, is important since it uniquely characterizes a bus route.

Boarding/Alighting and Bus Stop Identification: The detection of boarding (and alighting) on the bus can be done through observing the acceleration reading in z-axis (vertical axis). The key idea is, the vertical acceleration inside a bus, due to the jerking, is much higher than while walking or standing, which commuter is supposed to do either before or after the trip. Evidently, in Figure 4 we notice the discrepancy in vertical acceleration for the three positions – (i) when the user boards a bus, (ii) when she travels in the bus, and (iii) when she gets down from the bus. In Figure 4(a), the user boards the bus at 100 seconds after walking for a while, hence, at that point, we observe a sudden kink. When the user is inside the bus, the vertical acceleration diminishes sometime in between, indicating that those are the bus stops, as shown in Figure 4(b). Finally, as she gets down from the bus at a stop at near 1280 sec, indicated in Figure 4(c), the kink disappears.

Detection of Landmarks: We concentrate on the following five types of landmarks in a route (a) turns, (b) speed breakers, (c) bus stops, (d) bad roads, and (e) congestion zone. In *CrowdMap*, we primarily follow the proposed methodology in [4] to detect the landmarks in a route. The core concept is, we leverage on the smart-phone inertial sensor readings to detect the landmarks. For instance, congestion is characterized by the nature of halts; the sequence of stops and moves within a short duration (say 1 minute) is classified as a congested route segment.

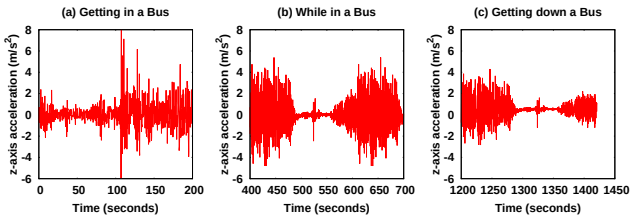

Figure 4: Vertical acceleration values to identify the events when the user is traveling by a bus: (a) the user boards in at 100 secs, (b) she is inside the bus when the bus takes a stop at 500 secs, (c) she gets down from the bus at 1280 secs

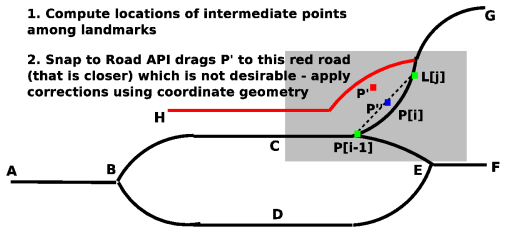

Figure 5: Trajectory generation procedure

2.2 Data Processing Module

This server-side module of *CrowdMap* processes the landmark data to extract the bus route & traffic information and generate path between two landmarks to generate the complete travel trajectory of the user.

2.2.1 Generation of Travel Trajectories

CrowdMap generates the travel trajectories from the sequence of opportunistic GPS annotated landmarks. The algorithm can be divided into two steps; the first is estimating the approximate coordinates of intermediate points from the location of landmarks, and the second is eliminating the estimation error.

Estimate location points: The objective is to estimate the location of the intermediate points from the landmarks. Let the sampling rate of the compass be $\mathcal{S}_C$. We compute the location of every points corresponding to the sampling points of the compass. The sequence of these points define the travel trajectory.

We utilize the fact that the sample rate of compass, $\mathcal{S}_C$, is very high, usually around 30-40 samples per second. Therefore, the distance traveled in this period ($1/\mathcal{S}_C$ sec) is very small and we can use the average speed of the bus v, computed from accelerometer readings by integrating over the acceleration value, to calculate this as $v \times 1/\mathcal{S}_C$ without incorporating any visible error. We thus have the distance traveled, angle of deviation (obtained from the list of sampling points from compass data indicating the angle of deviation at those points) and the previous location coordinates, which can be used to calculate next location coordinates with the help of Vincenty's formula [5]. However, due to the estimation methods for Vincenty's formula[5], every point may associate a small error which may further add up with a considerable deviation from the actual travel trajectory. In the next phase, we remove this estimation error.

Removing Estimation Error: Because of the cumulative error, it is possible that the estimated point goes out of the roads given in the actual map interface, like Google map interface. We can use Google's *Snap-to-Road API* to drag the point on the road. However, as the API projects a point over the nearest road segment, there is a possibility that the point is more closer to a different road where it is dragged into. Therefore, we use a coordinate geometric approach to drag the point at correct direction, as shown in the highlighted area of Figure 5. Let $P[i-1]$ be the previous estimated point on the route, P' be the current estimated point and $\mathbb{L}[j]$ be the next landmark in $\mathbb{L}$ succeeding point $P[i-1]$. Now, the new point should lie between $P[i-1]$ and $\mathbb{L}[j]$ on the route. Hence, we project P' on the line joining $P[i-1]$ and $\mathbb{L}[j]$ to get P''. We then apply the Google *Snap-to-Road API* on P'' to project the point on the correct road. The projected point $P[i]$ becomes part of the travel trajectory.

3. EVALUATION

We conducted experiments for a period of three months in the cities of Kolkata (KOL), Bhubaneswar (BBS) and Durgapur (DGP), three cities in the eastern part of India. While KOL and BBS are two state capitals with areas 1887 sqkm and 422 sqkm respectively, DGP is a suburban city with area 154 sqkm.

3.1 Experimental Setup

The application was distributed amongst 30 users. The first month of the experiments involved logging the complete GPS information of these users, so that we can later use this information as ground truth. Also, users were asked to tag the mode of transport they were using and the landmarks encountered. We conducted the experiments in 11 different routes of the above-mentioned cities and collected more than 200 trails.

3.2 Evaluation: Intelligent Logging

From the experimental traces, we check whether unnecessary data were collected when the mobile phones were stationary,

We checked all the data collected from the users using continuous GPS polling to observe the times when the GPS coordinates were within a 2 m radius for more than 5 *minutes* (indicating that the user is stationary), but still data were being collected. We observed that for only 0.08% of the readings, the system logged data even when the user was not in motion.

3.3 Evaluation of Detection of Road Features

We show two results here. First we give the error in the coverage of bad road patches followed by the results on congestion identification. The results related to detection of speed breakers, turns and bus stops is available in our work [4].

Ground Truth Generation: The ground truth data for detection bad road patches were collected from the data that the users had tagged during the first month of trail collection. The users were given a separate application where they tagged the coordinates of a bad road patch, whenever encountered. We employed a different strategy for collecting ground truth data for congestion. We took the help of Google map to see which parts of the routes were moderately or highly congested during the time of data collection, if the data were collected either in the morning or in the evening.

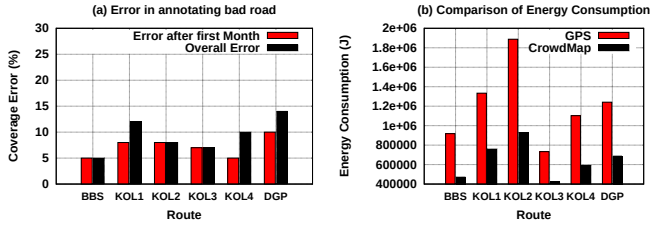

Figure 6: Bad road patch detection accuracy and Energy consumption analysis

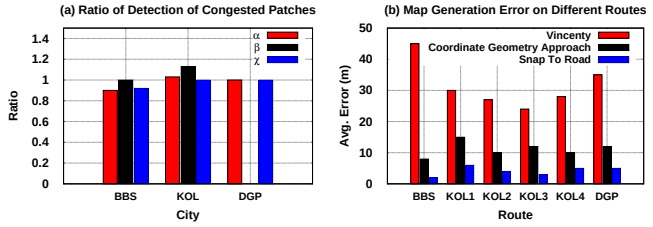

Figure 7: Evaluation of Congestion and Map generation module (a)Detection of detection of congested patches, (b) Travel trajectory estimation accuracy

Detection of Bad Road Patches: Figure 6(a) shows the average fraction of error in annotating the bad road patches. We compare it for two datasets - (1) data for the first month in which the tagging of bad road patches was done, and (2) on the entire dataset collected for three months. The intuition is that for the following months, the number of bad road patches may change. We observe that for some routes, the number of bad road patches do change, and hence the change in error rate. Considering accuracy, we see that the error in some cases is as high as 0.14. On investigating, we find that bad road patches in those cases correspond to one single lane of a road, and bus drivers bypass them whenever possible.

Detection of Congestion Affected Areas: We give three metrics to evaluate the accuracy of detecting congestion affected areas.

$$\alpha = \frac{\lambda_R^{det} + \lambda_Y^{det}}{\lambda}, \beta = \frac{\lambda_R^{det}}{\lambda_R^{act}}, \chi = \frac{\lambda_Y^{det}}{\lambda_Y^{act}}$$

where, $\lambda_{R/Y}^{det}$ is the length of a route tagged highly/medium congested by our system, $\lambda_{R/Y)}^{act}$ is the length of route found highly/medium congested from Google map, and λ is the total length of the congested route. Figure 7(a) gives the values for α, β and χ. A value greater than 1 implies that an extra stretch of the route was marked as congested. In case of DGP, there was only medium level congestion observed. However, we see that the error never shoots beyond 10% compared to what Google captures with an elaborate technology.

3.4 Evaluation: Travel Trajectory Estimation

We next evaluate how accurately are the estimated travel trajectories. We find out the localization error for the estimated GPS coordinates by comparing with ground truth GPS data. As mentioned, we already have the set of actual GPS coordinates for a route from the first month data. We compare the estimated GPS coordinates with the original GPS coordinates to find out the localization error. For this, we compute the average deviation between the estimated GPS coordinates and their closest GPS coordinates on the actual route as obtained from the ground truth data. In Section 2.2.1, we mentioned three stages in the estimation of a coordinate – first, by applying Vincenty's formula, followed by a coordinate geometry approach, and finally using Google *Snap-to-Road* API to reduce the estimation error. Figure 7(b) shows the localization error for these three stages on different routes. It can be easily observed that at each level, the error is considerably decreased, especially applying the coordinate geometry approach after Vincenty's estimation. Moreover, the maximum localization error never goes beyond $6m$.

3.5 Evaluation: GPS Usage

As expected, our approach consumes much less energy. As shown in Figure 6(b), the energy consumption without continuous polling of GPS is reduced down by 80% in majority of the cases.

4. CONCLUSION

One of the major motivations of this study is to design a low-cost solution that can gather useful information regarding road and route condition from users commuting in public transport. We design and implement an Android based *crowdsourcing* application named *CrowdMap*, which automatically identifies unique road signatures (like potholes, bad road patches) at zero user intervention. With a 3-month study, we observe that *CrowdMap* can detect these landmarks with an average accuracy of 95% and embeds the route segments on real map with an error of 6 meters at worst case. This study is a first step towards developing a public transport recommender system based on users preference.

Acknowledgement

The authors would like to thank Information Technology Research Academy (ITRA), Government of India, for supporting this work under the "DISARM" research project, sanction letter number and date ITRA/15(58)/MOBILE/DISRAM/01, Dt. 19-09-2013.

References

[1] IBM Global Commuter Pain Survey (available online, last accessed: June, 2016), 2011. http://www-03.ibm.com/press/us/en/pressrelease/35359.wss.

[2] J. Biagioni, T. Gerlich, T. Merrifield, and J. Eriksson. Easytracker: Automatic transit tracking, mapping, and arrival time prediction using smartphones. In *9th ACM Sensys*, pages 68–81, 2011.

[3] S. Nawaz and C. Mascolo. Mining users' significant driving routes with low-power sensors. In *12th ACM SenSys*, pages 236–250, 2014.

[4] R. Verma, A. Shrivastava, B. Mitra, S. Saha, N. Ganguly, S. Nandi, and S. Chakraborty. UrbanEye: An outdoor localization system for public transport. In *35th IEEE INFOCOM*, 2016.

[5] T. Vincenty. Transformation of co-ordinates between geodetic systems. *Survey Review*, 18(137):128–133, 1965.