Smart-phone based Spatio-temporal Sensing for Annotated Transit Map Generation

Rohit Verma, Surjya Ghosh, Niloy Ganguly, Bivas Mitra, Sandip Chakraborty
Indian Institute of Technology, Kharagpur, INDIA 721302
{rohit.verma,surjya.ghosh}@iitkgp.ac.in,{niloy,bivas,sandipc}@cse.iitkgp.ernet.in

ABSTRACT

City transit maps are one of the important resources for public navigation in today's digital world. However, the availability of transit maps for many developing countries is very limited, primarily due to the various socio-economic factors that drive the private operated and partially regulated transport services. Public transports at these cities are marred with many factors such as uncoordinated waiting time at bus stoppages, crowding in the bus, sporadic road conditions etc., which also need to be annotated so that commuters can take informed decision. Interestingly, many of these factors are spatio-temporal in nature. In this paper, we develop *CityMap*, a system to automatically extract transit routes along with their eccentricities from spatio-temporal crowdsensed data collected via commuters' smart-phones. We apply a learning based methodology coupled with a feature selection mechanism to filter out the necessary information from raw smart-phone sensor data with minimal user engagement and drain of battery power. A thorough evaluation of *CityMap*, conducted for more than two years over 11 different routes in 3 different cities in India, show that the system effectively annotates bus routes along with other route and road features with more than 90% of accuracy.

CCS CONCEPTS

•Information systems →Spatial-temporal systems; •Human-centered computing →Smartphones;

KEYWORDS

Spatio-temporal sensing; Map generation; City transports

ACM Reference format:
Rohit Verma, Surjya Ghosh, Niloy Ganguly, Bivas Mitra, Sandip Chakraborty. 2017. Smart-phone based Spatio-temporal Sensing for Annotated Transit Map Generation. In *Proceedings of SIGSPATIAL'17, Los Angeles Area, CA, USA, November 7–10, 2017,* 11 pages.
DOI: 10.1145/3139958.3140005

1 INTRODUCTION

Smartly annotated transit map [10, 19] is an important facility for navigation planning and assistance, which is useful for commuters, transit operators, city transport authorities as well as various government agencies. In recent times, Google transit provides a nice visualization of transit mapping system for various cities throughout the globe, however only limited towards the cities of developed countries [2][1]. Close investigations reveal that extending the transit map systems for developing countries face multiple challenges, broadly classified into two major areas – (i) transit system related challenges and (ii) infrastructure related challenges.

Challenges in transit system: (a) In most of the cities of developing countries, the public transport systems are usually managed through a public-private partnership model [4, 27]. This introduces a competitive environment for profit, where multiple owners run buses between the same pair of locations following different bus routes [27], which are uncoordinated and may have different features like stoppage profile and travel time. (b) Moreover, the types of buses may also vary widely; for instance, air-conditioned (AC) vs non air-conditioned (non-AC) buses; mini-buses vs normal buses etc [6]. (c) the bus stoppages and congestion in a route are not always fixed; buses may provide additional stoppages during busy hours based on passenger demand, and skip stops during the non-busy hours [24]. This (dynamic) information needs to be correctly annotated on the transit map, as these factors may severely impact the navigation planning of the commuters as mentioned in the Commuter Pain Survey by IBM [1]. **Challenges in road infrastructure:** (a) In developing countries, road infrastructure is challenging due to the presence of bad road patches, frequent speed breakers, sharp turns and congested areas, which impact the travel time and the comfort of the commuters. A transit map should annotate such eccentricities on the routes, such that commuters can choose the desired route while planning. (b) The aforesaid road information needs to be comprehensively collected in seamless manner, without relying on a fixed group of volunteers. (c) Since the condition of road changes over time, the information should be collected in real time, to cope up with this dynamicity.

Development of a scalable system to generate city transit map requires (i) automatic discovery of the public bus routes in a city, (ii) extracting their road and route specific characteristics, and (iii) rendering this information correctly on the discovered routes. Notably, rapid penetration of smart-phones equipped with onboard sensors opens up an opportunity to collect rich spatial data from the daily commuters in a city. We may rely on *crowdsensing* to populate the information repository containing sensor log collected from smart-phones of the commuters. Deep analysis of the collected sensor signatures reveals several road and route specific characteristics (say potholes, speed breakers, bus-stops etc), termed as *Point of Concerns* (PoCs). One may leverage on these PoCs to develop a

SIGSPATIAL'17, Los Angeles Area, CA, USA
© 2017 ACM. 978-1-4503-5490-5/17/11...$15.00
DOI: 10.1145/3139958.3140005

[1] Google transit is available only in 1.1%, 2.7% and 11.5% cities of Africa, South America and Asia, respectively, in contrast to 75.9% of the cities of North America

Table 1: Comparing *CityMap* with respect to different existing works, which focus on road surface monitoring and route map generation

Study Name	Brief description	Crowd-sourcing	Anomaly Detection	Machine Learning	End to End Route map generation	Map Annotation	Map suitable for public transport?
UrbanEye [24]	An energy-efficient, outdoor localization system for route navigation and travel time prediction for the city commuters	No	Yes	No	No	No	NA
Wolverine [7]	Identifies frequent braking events, which indicate congested traffic conditions and bumps on the roads to characterize the type of road using smart-phone sensors	No	Yes	Yes	No	No	NA
Dejavu [5]	A system that uses standard cell-phone sensors to provide accurate and energy-efficient outdoor localization suitable for car navigation.	Yes	Yes	No	No	No	NA
EasyTracker [8]	An automatic system for low-cost, real-time transit tracking, mapping and arrival time prediction	No	No	No	Yes	No	Yes
Mining driving routes [15]	Uses solely the accelerometer and gyroscope of the userfis phone to detect repeated driving routes.	No	No	No	Yes	No	No
Anomaly detection[20]	Explores the possibility of road anomaly detection via motorcycle-based mobile device using supervised and unsupervised machine learning techniques	No	Yes	Yes	No	No	NA

methodology to discover the bus routes from the spatio-temporally crowdsense data, extract the route specific features (say congestion in a route, possibility of getting a place to sit, average waiting time at a bus stop etc) and annotate the bus routes with these information. This paper takes an important step towards this direction.

In this paper, we develop *CityMap*, a crowdsensing based transit system, which can seamlessly sense the road & traffic condition of a city and render the route transit information. The architecture of *CityMap* is composed into two major modules – (a) database generation from crowdsensed repository, and (b) route discovery. In principle, smart-phone sensors provide unique signature to identify the PoCs such as speed breakers, turns, congestions etc to populate the database. However in practice, variability in sensor recording for different types of vehicles makes the detection methodology challenging. For example, the accelerometer readings vary for the same speed breakers, if recorded from a mini-bus vis-a-vis from a normal bus. Moreover, problem may arise due to significant battery consumption while continuous sensing is done to build up the crowdsensed repository (§3). In *CityMap* (*database generation* module), we implement a smart crowdsensing technique to collect sensor data from smart-phones only when commuters travel in a bus. Additionally, we develop a machine learning based classifier to identify PoCs observed in a route, which is smart enough to dynamically adapt itself depending on the city, vehicle and the environment. In *Route discovery* module, we leverage on detected PoCs to discover the routes and annotate the route map. Route discovery is challenging since there can be multiple routes between the same location pairs, and also a commuter may not board and alight only at the terminal stops. We implement the following three steps to discover routes – (a) identifying route trajectories from PoCs, (b) constructing route segments from trajectories, (c) stitching of route segments to generate a complete route (§4). Finally, we evaluate *CityMap* using data collected for 25 months from three different cities in India, and demonstrate that *CityMap* performs 65% better than the competing approaches, while *CityMap* consumes less battery power (§5). Delving deep, we exclusively evaluate individual components of *CityMap* demonstrating how correctly we can identify the PoCs, construct the route segments and finally stitch the segments to discover and annotate the complete routes(§6).

2 RELATED WORKS

Mobile sensing has been used in several applications related to user activity monitoring and to collect movement statistics of pedestrians as well as while traveling through vehicles. Several works[23, 28] rely completely on GPS, while a number of studies [7, 13, 24] have utilized inertial sensors in smart-phones, like accelerometer, gyroscope, compass etc., to identify unique road signatures. [5, 9, 11, 15] use these inertial sensors for navigation purposes in the absence of GPS. Works like [8, 12] generate route maps from smart-phone data or opportunistic GPS traces. Our work [22] proposes a *crowdsource* based solution, *CrowdMap*, which seamlessly collects travel data and generates the trajectory followed by the user.

Nevertheless, the limitations of these existing works are – (i) they primarily rely on war-driving data, and so there is significantly less possibility of noise, (ii) the developed mechanisms are limited to particular scenarios of smoothed data (like taxi movement in a city of a developed country), and (iii) uncertainty is not prevailing in city scale, and therefore combining the existing solutions of different subproblems (like inferring road conditions [15] or producing route maps [12] etc.) does not lead to the end-to-end solution in the scenario of a developing region. A summary of the existing works, which are closely related to the different modules of *CityMap*, is given in Table 1.

3 MOTIVATION, OPPORTUNITIES AND CHALLENGES

The prime motivation of this paper stems from the inherent limitations of Google transit information; we observe this facility absent for most of the cities across the globe. For instance, only 13 cities in India have Google Transit information available [2] against almost 495 cities. Notably, this coverage is pretty low for Asian, African and South American cities. One major reason behind this fact is that, the public transport in most of these cities are not centrally organized, rather privately owned, and hence, it is difficult to systematically gather information about all these routes [14, 17]. We explore the development of a *crowdsensing* based application which a commuter can install in her smart-phone. While commuting in a bus, this app can seamlessly log the information about the routes and update the transit information. In this solution, effectively no centralized system is required as it leverages on the *crowdsensing* to collect spatial data with transit information annotated. Distributing this application to a wide number of users would help to build the transit maps very fast.

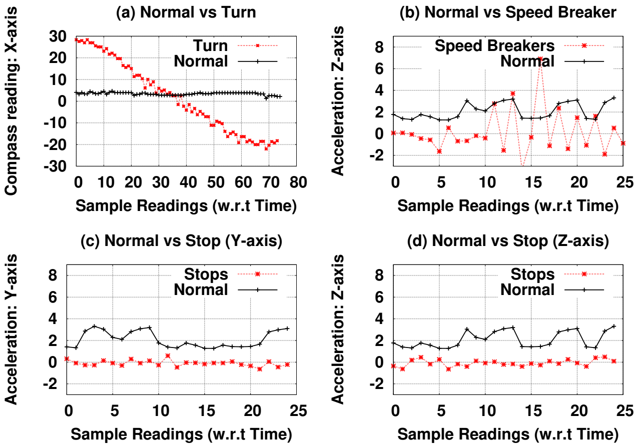


Figure 1: Every PoC registers unique signatures.

3.1 Opportunities for route discovery

Close inspections reveal that discovery of a new routes and discrimination between a pair of routes rely on the identification of *Points of Concerns (PoC)* present on a specific route. We define these *Points of Concerns (PoC)* of a route with the help of the features such as speed breakers, turns, bus stops, which render route specific signatures on smart-phone sensors. This can be observed from Fig 1, where compass reading changes sharply for turns and for speed breakers there is high peak in accelerometer's z-axis. As for stops, we observe the changes in acceleration along y and z-axis. Zero acceleration along y-axis and negligible variation along z-axis is tagged as a signature for bus stop. Based on the sensor log collected from the commuters, a classifier can be designed to uniquely identify these *Points of Concerns (PoC)* from the aforementioned sensor signatures. Once identified, the PoCs on a route can be put together to generate the complete trajectory with some approximation and matched to a route. Hence, if a trajectory has a similar set of PoCs, it can be linked to a bus route which has similar set of PoCs at similar coordinates.

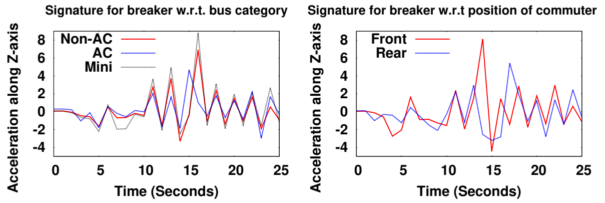


Figure 2: Impact on speed breaker signature in different scenarios

3.2 Challenges

However, the aforesaid approach comes with several challenges linked to it. We start with the challenges linked to crowdsensing. (a) It is difficult to collect crowd data if the application requires user intervention, or hogs the smart-phones battery power. In such scenarios, the commuter would rather prefer not to install and run the application itself. Moreover, the system should intelligently decide to collect data when commuter embark the bus and stop logging after alighting. (b) The detection of PoC(s) has challenges of its own. We may fix a set of static thresholds to identify the PoCs, however these thresholds would vary between cities. (c) Existence of different categories of buses also adds up to the challenges, as each of these also show variable signatures for the PoCs. This is evident from Fig 2(a), where we show the signature observed from different types of buses at the same speed breaker. Evidently, the high peaks indicate the position of the speed breaker. Nevertheless, one can notice that the range and characteristics of this peak varies across different categories of buses (say mini bus and AC bus). Additionally the position of the commuter in the bus also affects the sensor signatures as we have shown in Fig 2(b). (d) Laying down the PoCs to generate the trajectory is quite straightforward, but it is also required to estimate the intermediate points as these PoCs would be far apart usually. (e) Mapping of the trajectories with already discovered routes results in the construction and updating of routes. While linking the trajectory to a route, it is possible that two routes have overlapping segments. For instance, consider two bus routes $R1 : A - B - C - D$ and $R2 : A - B - E - F$, and a user has discovered a trajectory $A1 - B1$ similar to the route segment $A - B$. The question arises, should we link the trajectory $A1 - B1$ to route $R1$ or route $R2$. One possibility is to use features like speed of the bus, jerkiness of the bus etc. to characterize the routes and match the trajectories to the route which have better match. However as can be seen in Fig 3, there is no distinct characteristics to be identified leveraging directly on the amplitude of these features. (f) Moreover, these features vary both temporally and spatially which needs to be addressed too. (g) Finally, commuters do not always travel the complete route. In those cases, the disjoint trajectories should be stitched together to generate the complete route.

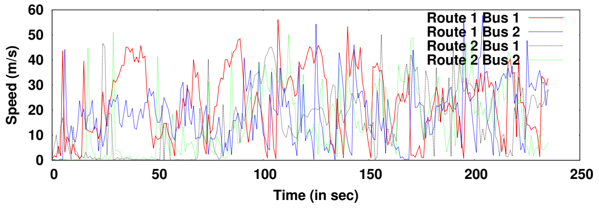


Figure 3: Variation of speed over two different routes for two different buses

4 SYSTEM DESCRIPTION

The overall system architecture of *CityMap* can be broadly divided into two modules – (a) Database Generation, and (b) Route Discovery, as shown in Fig. 4. The data is collected through *CityMap* smart-phone app and transferred to a remote database server. Based on the collected data, the *CityMap* server-side system generates the annotated transit map that can be rendered through both smart-phones as well as web applications.

4.1 Database generation

The rich database is the core part of the system. In order to construct the spatio-temporal database for transit system, we rely on

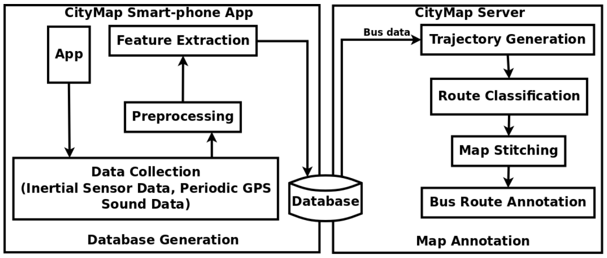


Figure 4: CityMap System Architecture

smart-phone based crowdsensing. We develop a data collection module, which accomplishes the following tasks – (a) intelligently identifies when to start and stop data logging, which can help to automatically start the data collection by detecting boarding and alighting of a passenger; (b) logs the sensor data from commuters' smart-phones, detects signatures to identify PoCs on the fly, and tags them with GPS coordinates; and (c) Identifies the route specific features for discrimination, considering the spatio-temporal aspects of the features.

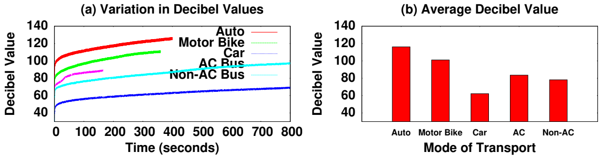


Figure 5: Variation of sound data for different types of motorized vehicles [22]

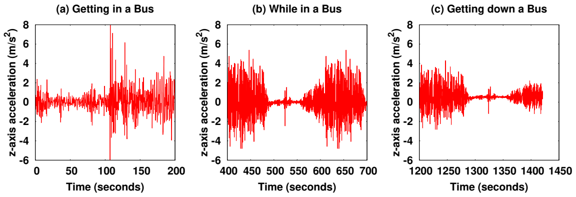


Figure 6: Accelerometer readings when user is boarding, in a bus and alighting from the bus [22]

4.1.1 ***Crowd-sourced data collection.*** We implement a specialized application by substantially extending our earlier work [22] with better adaptability for different cities and various types of vehicles. This smart-phone application is distributed amongst general commuters for route data collection. This application is non-intrusive as the commuter need not have to engage with the application. For instance, data sampling takes place only when the commuter is inside a bus, which the application can intelligently identify. As can be observed in Fig. 5, buses generate distinctive audio signatures (reflected in average decibel values) and hence are easily distinguishable. Moreover, leveraging on the distinct accelerometer signatures shown in Fig. 6, we can uniquely identify when to start and stop the data logging. Next, we demonstrate the automatic detection of *PoCs* (speed breakers, turns, bus stops) and the subsequent annotation of PoCs with the GPS co-ordinates. Precisely, GPS is only switched on when a PoC is detected.

Detection of PoCs: We develop a support vector machine (SVM) based classifier to identify the PoCs observed in a route. We learn the features (Table 2) to detect speed breaker, turns and bus stops. Notably, detection of PoCs simply based on the static thresholds (as shown in Fig. 1) is difficult under automated crowdsensing. This threshold varies across different cities and even between buses as shown in Fig. 2. Hence, the classifier should be smart enough to dynamically adapt itself depending on the city, vehicle and the environment, which is the motivation behind using SVM in this case. In this classifier, we leverage on the mean and standard deviation of z-axis acceleration along with a peak function, which looks for available peak as shown in Fig. 1, to detect speed breakers. As can be seen in Fig. 7(a), mean and standard deviation show a distinct characteristic. However, we have included the peak function to mitigate the false positives that can occur due to the driving behavior. Similarly, we rely on the mean and standard deviation of compass readings for detecting turns. It can be observed from Fig. 7(b) that these features are quite discriminating in identifying turns. In order to detect bus stops, we compute the mean and the standard deviation of z-axis and y-axis acceleration. It is important to note that the training of the classifier is done offline at a remote *CityMap* server, and the trained SVM module is periodically updated to the smart-phone app for online classification. Hence, the SVM based classification wouldn't slow down the application.

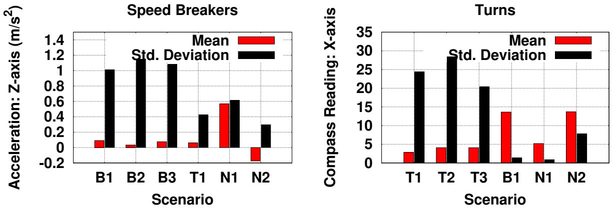
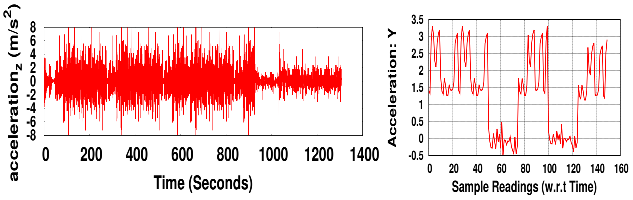


Figure 7: (a) Z-axis accelerometer reading for detecting a speed breaker: B1, B2 and B3 - same speed breaker, different buses, T1 - turn, N1, N2 - no PoCs; (b) Compass reading for turns

Table 2: Features used to classify PoCs from sensor data (μ : mean, σ : standard deviation, $peak_{accz}$: peak observed in z-axis acceleration, com : compass readings; The sample size for mean and standard deviation calculation is of 25 readings)

PoC	Features
Speed Breaker	$\mu_{accz}, \sigma_{accz}, peak_{accz}$
Turn	$\mu_{com}, \sigma_{com},$
Stops	$\mu_{accz}, \sigma_{accz}, \mu_{accy}, \sigma_{accy}$

Once the PoCs are identified, the smart-phone app transfers the geo-tagged PoC data to a remote *CityMap* server. We then process the data at the *CityMap* server to identify the annotated bus routes based on the route specific features, as discussed next.



(a) User in a bus gets seat at 900s **(b) Congestion signature**

Figure 8

4.1.2 *Feature identification for bus route characterization.*

Each route R can be uniquely characterized with the help of a suite of features F_R. In the following, we propose the features to annotate the route map as well as to distinguish the bus routes.

Speed of the vehicle (v): Let $a(t)$ be the acceleration reading at time t. Then, the average speed of the vehicle is calculated as $v = \int_0^1 a(t)dt$. This value is re-calibrated every time a GPS polling is done. Buses follow different speed at different routes, hence it can be used as a discriminating signature.

Speed before approaching a PoC (v_{lm}): We compute this features as, $v_{lm} = \frac{1}{5}(\sum_{n=10}^{15}(\int_n^{n+1} adt))$, which depicts the average speed of the vehicle 10s before approaching a PoC, calculated over 5 samples. The speed with which a bus approaches a PoC is crucial in many scenarios, like when taking a turn, a very high speed would be dangerous.

Average waiting time at a bus stop (t_w): Assuming wait time at a bus stop as t, we compute average waiting time $t_w = \frac{1}{|trips|}(\sum_{trips} t)$, where $|trips|$ represents the number of trips. Different buses have different wait time at the same bus stops, hence this feature can characterize a route.

Probability of sitting (p_{sit}): Fig. 8a shows the signatures when the user is standing for sometime after getting in the bus, and then gets a vacant seat to sit down. It is easily visible that when the user is standing, the acceleration values are more pronounced with multiple peaks and dips. We tag a trip as $trip_{sat}$ if the user gets a seat, then $p_{sit} = \frac{|trip_{sat}|}{|trips|}$. This feature is important because getting a seat is a major concern for many passengers [1].

Jerkiness of the bus (J): Jerk is given as $da(t)/dt$ and $jerk_{critical}$ is when $jerk \leq -9.9m/s^3$[18], within a sampling window of 5s. We compute jerkiness as, $J = \frac{|jerk_{critical}|}{|samples|}$ Jerkiness is an important feature as it helps in determining the road condition as well as driving behavior.

Probability of skipping a bus stop (p_{skip}): We label a stop as S^i, where i is the stop number, then $p_{skip} = \frac{|S^i_{skipped}|}{|trips|}$. Many commuters would be concerned whether the bus will skip there stop, especially in crowded buses, thus we include this feature in our study.

Congestion (C): Congestion has a continuous stop-move-stop-move pattern, which we detect using accelerometer readings, as shown in Fig. 8b. The region close to zero are the ones when the bus is not moving, and the other are when it is mobile. However these have peaks because of the sudden brakes. Let the time period for the stop-move pattern be t_{sm}, then we have medium congestion when $1min \leq t_{sm} < 5min \rightarrow C_{high}$ and high congestion when $t_{sm} \geq 5min \rightarrow C_{high}$

In a nutshell, a route R can be represented as the feature suite $F_R = \{t_w^R, p_{sit}^R, v^R, v_{lm}^R, J^R, p_{skip}^R, C^R\}$.

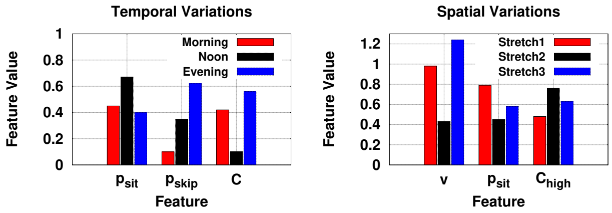


Figure 9: (a) Temporal variations; p_{sit} at a popular bus stop, p_{skip} for a school bus stop, C: length of congested route segments/total route length (b) Spatial variations: v: avg. speed of the route in the stretch/average speed of the bus over the route, p_{sit} in the stretch, C_{high}: highly congested route/total congested route

4.1.3 *Capturing temporal and spatial variations.*

Temporal variation: Fig 9 demonstrates the fact that the traffic in a route varies widely at different times of the day, depending on factors like congestion, importance of a bus stop, probability of sitting etc. Hence a gross aggregation (in terms of average) of the aforesaid features F for a route will lead to highly erroneous signatures. In order to capture this temporal variation, we split the day into multiple time zones, (i) the busy hours (7am to 12pm and 5pm to 9pm) and (ii) the remaining non-busy hours. We calculate the features in F for each of the time zones separately while populating the database and classifying routes.

Spatial variation: A similar variation is observed in different route segments too, which is evident from Fig. 9(b), where we show the variation in features in three stretches at the start, an intermediate stretch and close to the end of the route. Calculating the feature value averaged over the complete route would never be a proper measurement of route features lest we divide the complete route into multiple segments. We first divide the segments between any two set of PoCs. After this segmentation, if some segments are larger than 3km, we again divide them into smaller segments of approximately equal lengths.

4.2 Route discovery

The detected set of PoCs can now be used to discover the routes and annotate the route map. In order to accomplish this task, the following steps need to be performed – (a) discovering route trajectories from PoCs, (b) mapping the trajectory with the existing bus routes to construct a route segment, (c) stitching of route segments to generate a complete route.

4.2.1 *Trajectory discovery.*

The system utilizes the sequence of GPS annotated PoCs to generate the travel trajectories. This is accomplished in two steps – (a) first, laying down the PoCs to construct the route segment, and (b) estimating the intermediate coordinates using Vincenty's formula [25]. However, this estimation adds some error which could get accumulated to give unusable estimations. We thus introduce an error correction technique following

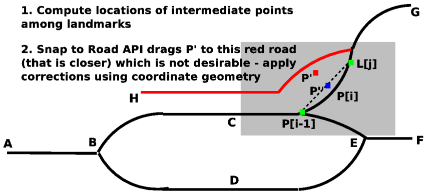

1. Compute locations of intermediate points among landmarks
2. Snap to Road API drags P' to this red road (that is closer) which is not desirable - apply corrections using coordinate geometry
G
L[j]
P'
P''
P[i]
H
C
P[i-1]
A
B
E
F
D

Figure 10: Trajectory generation procedure

coordinate geometry principle and Google Snap-to-Road API [3]. Consider Fig. 10, P[i-1] is the last estimated location and L[j] is the next PoC. Using Vincenty's formula, the point P is estimated. Here if we use Snap-to-Road, the point will fall on the road H. Hence, we project the point P on the line joining P[i-1] and L[j], and then use Snap-to-Road to estimate the point P[i] that lies on the target road. Once the trajectory is generated, the next task is to include this trajectory into one of the possible routes.

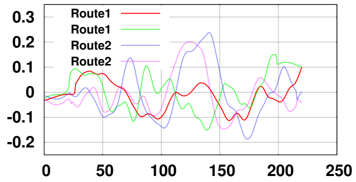


Figure 11: Variation of speed after DWT for different data

4.2.2 ***Constructing route segment: Inclusion of trajectory to a route.*** Considering the data obtained from the *individual commuter* in a controlled environment, embedding it on the map is a simple task of just listing down and tagging the map with the PoC information. A major issue arises when working in real environment with the collected *crowd-sourced data* for public buses with minimal GPS information; it becomes difficult for the system to identify the bus route unless explicitly tagged. This is important since in the next step, discovered trajectories need to be stitched together to generate the complete route, and this stitching relies completely on the disambiguation of trajectories. We employ *Discrete Wavelet Transforms* (DWT) to map each trajectory with the respective route. In principle, a feature, say speed of the bus, would have high correlation for the buses traveling in the same route while low correlation with the buses running in different routes. Observing this correlation is rather challenging in the amplitude domain, as shown in Section 3. We transform the features using DWT to capture the frequency domain insights, and then compute the correlation. In Fig. 11, we plot the variation of speed after performing DWT for a set of 4 data samples, 2 belonging to a single route. This result demonstrates the fact that the buses running in same routes are highly correlated with almost similar peaks and dips as well as variation. Following this principle, we employ a strategy to map a discovered trajectory x_{ij} to a specific route $\mathbb{T}$.

The trajectory x_{ij} can be represented with the help of the feature suite $F_x^{ij} = \{v^{ij}, v_{lm}^{ij}, p_{sit}^{ij}, p_{skip}^{ij}, J^{ij}\}$. Now, for each feature in F_x^{ij}, we compute DWT and estimate the *Pearson Coefficient* of the same for every route $\mathbb{T} \in \mathfrak{R}$, where $\mathfrak{R}$ is the database of routes. It is expected that if this trajectory x_{ij} is a part of a specific route $\mathbb{T}$, the correlation between x_{ij} and for the matching route $\mathbb{T}$ will be high. Precisely, we compute the correlation coefficient for all the routes in $\mathfrak{R}$ and select the one for which the correlation is highest. We claim that a trajectory x_{ij} is part of a route $\mathbb{T} \in \mathfrak{R}$, if for three or more features in F_x^{ij}, the calculated Pearson Coefficient between $\mathbb{T}$ and $\mathfrak{R}$ is greater than a threshold ξ. Once a discovered trajectory x_{ij} gets included as a part of a route $\mathbb{T}$, we call it as a *route segment* of $\mathbb{T}$.

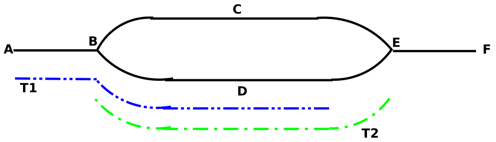


Figure 12: The trails T1 and T2 are from different users and need to be stitched together to get the route A-B-D-E

4.2.3 ***Map stitching for bus routes.*** Since commuters travel different segments of a bus route and may not travel the entire end to end route, it is essential to identify the different route segments accurately and combine them to construct the complete bus route. Consider the road network given in Fig. 12. A commuter may follow route segment $T1$ from point A to point D, while another commuter may go through segment $T2$ which is from point B to point E via point D. We aim to stitch segments $T1$ and $T2$ to get the complete bus route. Here we take a conservative approach, where we stitch two route segments only if they are overlapped, as in the case of Fig. 12.

Challenge and opportunity: Nevertheless, finding overlaps in route segments is not always sufficient, as there can be two different routes having overlap between them. Considering Fig. 12, assume there are two bus routes – $A–B–C–E–F$ and $A–B–D–E–F$. Now if a commuter travels from A to B only, the segment $A - B$ matches with both the bus routes. However, we may leverage on the fact that buses at different routes exhibit diverse route specific characteristics which can be captured in the features suite F of a route (say speeds, waiting time at the stoppages, probability of bus stops skipped etc), as introduced earlier. In this line, we propose the map stitching algorithm, as given in Algorithm 1.

Algorithm: The Algorithm takes input POC_x – the list containing coordinates of PoCs encountered in the route segment. It utilizes *RouteLog* table that contains details (POC_R, F_R) of every route R encountered so far. The algorithm runs in two phases – (1) *road matching* and (2) *route characteristics matching*. In the *road matching* phase, the route segment $\mathcal{T}$ is compared with all the bus routes in *RouteLog* to check whether they follow the same or an overlapping road segment. However, as the location points may not be exactly same due to GPS error, we apply *longest common subsequence* (LCSS) trajectory comparison algorithm to determine possible matching road segments [26]. In order to find the matching road segments, the number of common PoCs in POC_x and POC_R is computed An existing bus route $\mathbb{T} \in \mathfrak{R}$ is considered to be a candidate for matching road segment if the following conditions are satisfied – (a)

Algorithm 1: Map Stitching Algorithm

```
Input: RouteLog Table that contains details ($POC_R$, $F_R$) of every route $R$ encountered so
       far, $\mathcal{T}$ – the detected route segment, $\mathfrak{R}$ – the existing database of the routes
Output: Include the route segment $\mathcal{T}$ as a part of existing route or generate a new route –
        Update the route database $\mathfrak{R}$
/* Route matching                                                                  */
$\mathbb{D} \leftarrow \Phi$;
foreach $\mathbb{T} \in \mathfrak{R}$ do
    if $\exists$ common PoC subsequences $POC_{\mathbb{T}} \in \mathbb{T}$ and $POC_{\mathcal{T}} \in \mathcal{T}$ and
      LCSS-Distance($POC_{\mathbb{T}}$, $POC_{\mathcal{T}}$)$\leq \zeta$ then
        $\mathbb{D} \leftarrow \mathbb{D} \cup \mathbb{T}$;
/* Route characteristic matching                                                     */
if $\mathbb{D} \neq \Phi$ then
    Compute DWT for $[v, v_{lm}, p_{sit}, p_{skip}, J]$ of the route segment
    $\rho_{max} \leftarrow 0$ ;                                    /* temporary variable */
    $\mathbb{T}_{sel} \leftarrow \Phi$ ;                                /* temporary variable for route */
    foreach $\mathbb{T} \in \mathbb{D}$ do
        Compute DWT for $(v, v_{lm}, p_{sit}, p_{skip}, J)$ for $\mathbb{T}$
        /* Compute Pearson Correlation Coefficient between the DWT of $\mathcal{T}$
           and $\mathbb{T}$                                                       */
        $[PCC_v, PCC_{v_{lm}}, PCC_{p_{sit}}, PCC_{p_{skip}}, PCC_J] \leftarrow$ Pearson Correlation
          Coefficient for $v, v_{lm}, p_{sit}, p_{skip}, J$ between the trajectory and $\mathbb{T}$
        $count \leftarrow$ Number of features $i$ such that $PCC_i > \xi$ ; /* $\xi$ is a threshold
          on Correlation Coefficient */
        $topmean \leftarrow$ Mean of top 3 $PCC_i$
        if $count >= 3$ and $topmean > \rho_{max}$ then
            $\rho_{max} \leftarrow topmean$
            $\mathbb{T}_{sel} \leftarrow \mathbb{T}$
    if $\mathbb{T}_{sel} \neq \Phi$ then
        /* There is a match with an existing bus route                               */
        Update RouteLog table for route $\mathbb{T}_{sel}$ with averaging the features of $\mathcal{T}$ and
          $\mathbb{T}_{sel}$;
    else
        /* This is a probable new bus route -- keep it in data store for
           off-line processing                                                      */
        Add $\mathcal{T}$ to the NewRoute table along with bus features;
else
    /* A new road has been discovered -- keep it in data store for off-line
       processing                                                                   */
    Add $\mathcal{T}$ to the NewRoute table along with bus features;
```

there exists common PoC subsequences among $\mathcal{T}$ and $\mathbb{T}$, and the subsequence length is more than κ, and (b) the LCSS distance [26] among the common PoC subsequences of $\mathcal{T}$ and $\mathbb{T}$ is less than a threshold ζ, where we set κ and ζ values following [26]. The next step is determined based on the output of road matching. As mentioned earlier there can be more than one match, as the route segment $\mathcal{T}$ can be a part of multiple bus routes. If there is no match, then the route segment is included in an off-line database as a probable new bus route. Here we take a conservative approach to include new bus routes, as there is always possibility of noise within the *crowdsensed* data. We process this off-line database periodically, only if we get number of travel traces at a particular road more than a threshold. Next, we proceed to compare route specific characteristics as a matching route segment might be a completely new route. Route classification is employed to get the possible routes with which this segment matches. Out of this set of routes, we select the one for which the mean value of top three Pearson Correlation Coefficient is maximum. If no such route segment is identified, $\mathcal{T}$ is probably a new bus route, and we keep it for off-line processing. Otherwise, we identify the additional route segment between $\mathcal{T}$ and $\mathbb{T}_{sel}$ and add it to the map. At the same time, we update the location of common PoCs as well as the bus features between $\mathcal{T}$ and $\mathbb{T}_{sel}$, by computing the average of PoC locations and feature vectors. This gives us the complete set of annotated bus routes encountered so far by the *CityMap* application.

5 EVALUATION OF *CITYMAP*

In this section, we evaluate the overall performance of *CityMap*. First, we measure the elegance of *CityMap* in (i) correctly detecting bus routes and (ii) estimating the extent of mismatch between the detected route and the original route. Next, we show the performance of *CityMap* against a competing algorithm which is driven by hidden Markov model [16].

5.1 Experimental setup

We conducted experiments from March 2015 to May 2017 in the cities of Kolkata (KOL), Bhubaneswar (BBS) and Durgapur (DGP), three cities in the eastern part of India. The data is available online at [21]. We engaged 30 volunteers, who were all college students in the age group of 18-25 years, to conduct the experiment. We distributed the data collection application among all these 30 subjects. The first three months of the experiment involved logging continuous GPS information to construct the ground truth information. Additionally, subjects were asked to label the mode of transport they availed and the specific PoCs they encountered in the route. The devices employed in this experiment ranged from low-end Android devices (JellyBean to Marshmallow) to high-end ones. We conducted the experiments in 11 different bus routes of the aforementioned three cities. As a sample, detailed statistics of 6 bus routes (out of these 11) are shown in Table 3.

Table 3: Route details ($\mathcal{N}$: Route name – city in brackets, $\mathcal{L}$: Route length, $\mathcal{T}$: Daily avg. travel duration)

$\mathcal{N}$	$\mathcal{L}$ (km)	$\mathcal{T}$ (hour)	$\mathcal{N}$	$\mathcal{L}$ (km)	$\mathcal{T}$ (hour)
K1 (KOL)	17	3.12	K2 (KOL)	14	2.76
K3 (KOL)	20	4.32	K4 (KOL)	10	0.48
B (BBS)	19	1.20	D (DGP)	22	3.84

5.2 Performance of bus route detection

First, we evaluate the overall performance of *CityMap* by measuring the accuracy of bus route detection. We perform the comparison of *CityMap* against the ground truth information collected in the first three months period, where the volunteers tagged the bus number every time they traveled. We compute three classical parameters – precision (P), recall (R) and overall accuracy (A) to evaluate *CityMap*. *CityMap* has an average precision of 93%, 91% recall and accuracy of 87% in detecting the correct bus routes over all the cities. Table 4 summarizes these three parameters over two typical routes in Kolkata (KOL), Bhubaneswar (BBS), and Durgapur (DGP) chosen judiciously. We select bus routes $S9$ & $S4$ in KOL and $A1$ & $A2$, which have an overlapping patch of around 8 and 10 km respectively, and 306 & 207 in BBS which follow the same road throughout. The table shows that the detection accuracy is more than 90% in KOL and more than 80% in BBS and DGP.

5.2.1 ***Route mismatch fraction.*** Next, we investigate the extent of mismatch between the estimated route and the original route, by computing the length of non-overlapping portions between the two, normalized by total route length. For this, we measure *route*

Table 4: Accuracy of Bus Route Differentiation

City	Route Number					
	306			207		
BBS	P	R	A	P	R	A
	0.93	0.88	0.83	0.92	0.92	0.85
	S9			S4		
KOL	P	R	A	P	R	A
	1	0.9	0.9	0.96	0.96	0.93
	A1			A2		
DGP	P	R	A	P	R	A
	0.92	0.92	0.86	0.93	0.93	0.88

mismatch fraction (RMF) as defined in [16]. Let δ_m be the total length of the segment, which does not overlap with the actual route, and δ be the total length of the route. We compute RMF (γ) as, $\gamma = \frac{\delta_m}{\delta}$. In Fig. 13(a), we evaluate RMF for four routes in Kolkata and one route each in Bhubaneswar and Durgapur. The low RMF values for routes K2 and K3 can be linked to the higher landmark density. K2 and K3 have landmark density of $6.2/km$ and $5.78/km$, whereas the values for B, K1, K4 and D are $3.54/km$, $3.43/km$, $3.27/km$ and $3.82/km$

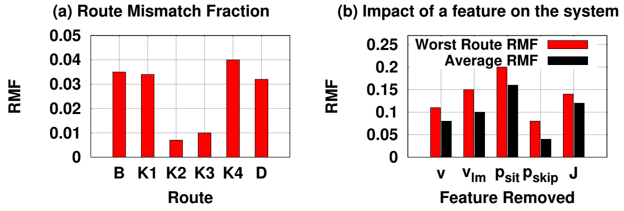


Figure 13: Determining Route Mismatch Fraction for the routes and Impact of a feature on the system

5.3 Competing heuristic

We evaluate the performance of *CityMap* against a baseline algorithm proposed in [16], which mostly relies on the hidden Markov model. The evaluation has been performed from the perspective of accurate route discovery and the cost paid in terms of energy drain.

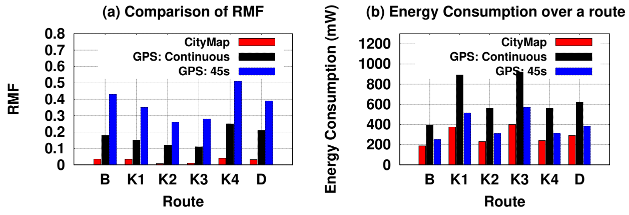


Figure 14: Comparison with the competing system

5.3.1 ***Route detection***. The competing heuristics only takes into account the time stamped GPS information and hence fails while classifying the overlapping segments of two routes (Fig. 14(a)). We have used two variations of the competing system. First is the case when continuous GPS sampling is done, and second is with a GPS sampling period of $45s$. As can be observed in Fig. 14(a), absence of the classification module in competing algorithm results in higher values of RMF, as many of the trajectories although correctly stitched are not classified to the correct route. In contrast, *CityMap* demonstrates an accuracy higher than 80%.

5.3.2 ***Energy consumption***. Fig. 14(b) illustrates the energy consumed by *CityMap* for different routes. We implement the following two baselines for comparison: (i) GPS being used throughout the route, which is the best case for the competing system and (ii) GPS sampled for every $45s$. It is evident that the energy consumption for *CityMap* is considerably low.

6 DISSECTING *CITYMAP*

In this section, we evaluate the two major components of *CityMap* (a) Database generation and (b) Route discovery. Next, we illustrate the importance of the crowdsensing on the performance of *CityMap* and also reveal the importance of features suite F_R.

6.1 Evaluation: Database generation

First, we show the accuracy of detecting PoCs (viz, speed breakers, turns and bus stops) and next, demonstrate the correctness of congestion identification. This evaluation is necessary to correctly discover the routes and annotate its eccentricities on the route map.

6.1.1 ***Ground truth generation***. As mentioned in the experimental setup, the ground truth data of the PoCs were collected from the information tagged by the subjects during the first three months of trail collection. Table 5 enumerates the ground truth PoC information for different cities. In case of congestion, we rely on Google map to check which parts of the routes were moderately or highly congested for two times of the day (morning or evening).

Table 5: Ground truth details from different cities. PoCs are in numbers and congestion patches are in Km.

City	Turns	Speed Breakers	Bus Stop	Morning (Km.)		Evening (Km.)	
				Medium	High	Medium	High
BBS	13	5	25	6	2	7	3
KOL	44	15	204	19	8	20	10
DGP	32	10	49	0.5	0	1.5	0

Table 6: PoC detection performance (P: Precision, R: Recall and A: Accuracy)

City	Turns			Speed Breakers			Bus Stops		
	P	R	A	P	R	A	P	R	A
BBS	0.93	1	0.93	0.94	1	0.94	0.93	0.9	0.84
KOL	0.98	1	0.98	1	1	1	0.97	0.96	0.93
DGP	0.97	0.97	0.94	0.93	1	0.93	0.94	0.96	0.91

Table 7: Comparison using classifier and static thresholds

City	Classifier			Average Static			Best Static		
	T	B	S	T	B	S	T	B	S
BBS	0.93	0.94	0.84	0.93	0.8	0.84	0.62	0.4	0.84
KOL	0.98	1	0.93	0.89	0.75	0.93	0.98	1	0.93
DGP	0.94	0.93	0.91	0.81	0.6	0.91	0.63	0.3	0.91

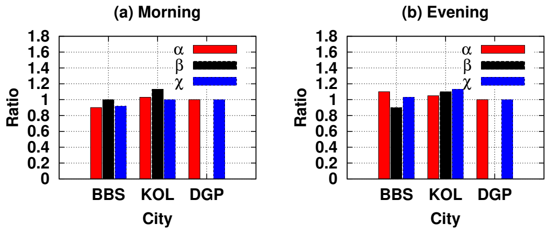


Figure 15: Ratio of detection of congested patches

6.1.2 ***Detection of PoCs.*** Accurate detection of PoCs is important for the correct route discovery. In Table 6, we present accuracy of the classifier in detecting various PoCs in different cities. We compute the precision, recall and accuracy values for all the three types of PoCs. It can be noticed that turns and speed breakers are easily detected with very high accuracy. Next, we evaluate the performance of the classifier against the simple baseline with static thresholds. Table 7 exhibits the accuracy of the PoC detection for three competing cases – (a) our proposed SVM based classifier, (b) considering the average of the (static) thresholds calculated for the three respective cities and apply this (average) threshold for detection of PoCs, and (c) only considering the static threshold which gives the best results in any one of the target cities and apply that to other cities (in our case, threshold for Kolkata). The approach of selecting the best threshold is common with competing endeavors like Dejavu [5] and Nericell [13]. We observe that the accuracy drops considerably for the static threshold based baselines.

6.1.3 ***Detection of congestion affected areas.*** We propose the following three metrics to evaluate the accuracy.

$$\alpha = \frac{\lambda_H^{det} + \lambda_M^{det}}{\lambda}, \beta = \frac{\lambda_H^{det}}{\lambda_H^{act}}, \chi = \frac{\lambda_M^{det}}{\lambda_M^{act}}$$

where, $\lambda_{H/M}^{det}$ is the length of a route tagged highly/medium congested by our system, $\lambda_{H/M}^{act}$ is the length of route tagged as highly/ medium congested from Google map, and λ is the total length of the congested route. Notably, in Fig. 15, we evaluate the detection accuracy for two times of a day; in morning and evening. Occasionally, we observe the value greater than 1, which implies that an extra stretch of the route was detected as congested.

6.2 Evaluation: Route discovery

In the following, we exclusively evaluate the individual steps of the route discovery module.

6.2.1 ***Accuracy of trajectory discovery.*** In Fig. 16(a), we exhibit the localization accuracy for trajectory discovery. We take the GPS data collected in the first three months as ground truth and compare the accuracy with respect to this. It is comforting for us to note that the error never goes beyond $6m$. It is quite evident that inclusion of error correction schemes following coordinate geometry and Snap-to-Road substantially improves the accuracy.

6.2.2 ***Accuracy of constructing route segments.*** Next, we investigate how accurately we can map and include the discovered trajectories with the respective routes and subsequently construct the route segments. Fig. 16(b) shows the fraction of trajectories which are erroneously not included in the correct route and Fig. 17(a) shows the average length of such erroneously classified trajectories. This is important to note that routes like K3 have lower error than route K2, but exhibits longer average error length. The errors observed in the finally discovered routes thus display a trade-off between these two values. Evidently, in Fig 17(a), we observe that the route K4 shows lower percentage error compared to D, however higher error in average length. Hence, eventually, D manifests lower RMF compared to K3.

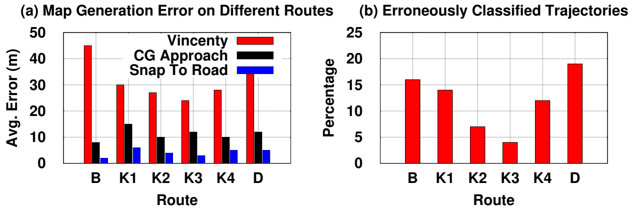


Figure 16: (a)Localization error for generated trajectories, (b) Percentage of trajectories erroneously classified

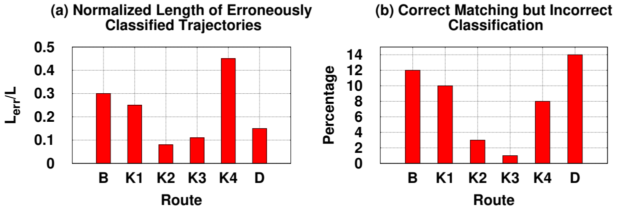


Figure 17: (a) Avg. length of erroneously classified trajectories – L_{err}: Avg. length of erroneously classified trajectories, L: Length of the correct candidate route; (b) Percentage of routes correctly stitched but erroneously classified

Selection of threshold ξ: We empirically choose the threshold ξ for classification of route segments. We claim that the classification is a *success* when the trajectory is correctly matched with an existing bus route. An *error* occurs if the trajectory is matched with a bus route where it is not a part of. We observed that both error and success decrease considerably when ξ is increased from 0 to 0.5. Error falls from 60% to mere 6%, whereas success falls to 80% from 100%. This decrease in success becomes higher with almost static error with further changes in ξ. We fix 0.5 for the threshold ξ, as we observe minimum error while attaining high success.

6.2.3 ***Accuracy of map stitching.*** Finally we evaluate the performance of the map stitching heuristics. The overall accuracy of *CityMap* does not always manifest the accomplishment of the map stitching heuristics. Since one may discover multiple overlapping route segments for different routes, there is a possibility that even if a trajectory is classified to a wrong route, as the segments overlap, the stitching is perfect. We calculate the percentage of such cases in Fig. 17(b) to exclusively evaluate map stitching. Comparing this result with Fig. 16(b), we observe that the stitching accuracy is linked to the accuracy of route classification. This is an important reason why *CityMap* fared well compared to [16].

6.3 Insights of *CityMap*

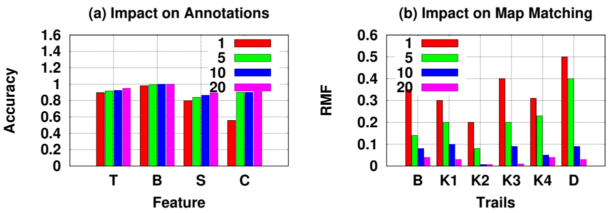


Figure 18: Impact of data accumulation: (a) Detection of road/route features (T: Turn, B: Breaker, S: Stop, C: Congestion) (b) RMF

6.3.1 ***Impact of data accumulation.*** The success of a *crowdsensing* application depends upon its performance improvement with peoples' participation. Fig. 18 manifests the fact that stretching the experiment period (aka gathering more volume of data) helps to improve the performance in the light of RMF. The results are shown for a trail collection duration of 1, 5, 10 and 20 days. It is evident that the accuracy of *CityMap* improves, and RMF reduces in all the cases as more data being added.

6.3.2 ***Feature analysis.*** In this experiment, we eliminate one of the features from suite $F = \{v, v_{lm}, p_{sit}, p_{skip}, J, \text{C}\}$ and run the algorithm to rank the features based on importance. We exhibit the results for two scenarios. (a) One is for the route for which we got the worst result, i.e. K4, because we believe the reason for the bad results could be amplified and (b) we compute the average RMFs of all the routes. We observe from Fig. 13(b) that p_{skip} affects the value of RMF minimum. This is rather intuitive because the non-popular bus stops are usually skipped in all the routes with few exceptions. p_{sit} impacts the result most, which can be linked to the fact that the probability of getting a seat is highly route specific. J and v_{lm} have similar impact almost as both of these features are linked to driver behavior and hence specific to the route the driver follows. v also doesn't have very high impact because majority of the buses have almost similar speed range.

7 CONCLUSION

In this paper, we have developed *CityMap*, a system to automatically extract spatio-temporal transit information from smart-phone sensor data and to annotate them on the route map. In order to build up this system, we have relied on crowdsensed data, and therefore, we have considered the factors like zero user engagements, minimal usage of smart-phone resources, extendability and robustness of the system over multiple cities, possibility of having noisy data, and the requirement to generate rich transit information for the commuters to take correct decision during travel. We have done a thorough testing of the system over multiple routes at multiple cities, and observed the system performance to be significantly better than the competing baselines. Apart from discovering the bus routes, *CityMap* also extracts various road specific features to characterize the comfort during a journey (say, sharp turns, bad road patches, frequent speed breakers etc.) and the route specific features (say, bus stoppages and their waiting times, crowding in a bus etc.), relying on which a commuter can decide the most suitable route among multiple possibilities to reach the destination.

REFERENCES

[1] 2011. IBM Global Commuter Pain Survey: Traffic Congestion Down, Pain Way Up (available online, last accessed: June, 2017). (2011). http://www-03.ibm.com/press/us/en/pressrelease/35359.wss.

[2] 2017. Google Transit Cities (available online, last accessed: June, 2017). (2017). https://maps.google.com/landing/transit/cities.

[3] 2017. Snap To Road (available online, last accessed: June, 2017). (2017). https://developers.google.com/maps/documentation/roads/snap.

[4] Sohail Ahmad and Jose A Puppim de Oliveira. 2016. Determinants of urban mobility in India: Lessons for promoting sustainable and inclusive urban transportation in developing countries. *Transport Policy* 50 (2016), 106–114.

[5] Heba Aly and Moustafa Youssef. 2013. Dejavu: an accurate energy-efficient outdoor localization system. In *21st ACM SIGSPATIAL.* 154–163.

[6] Bilge Atasoy, Takuro Ikeda, Xiang Song, and Moshe E Ben-Akiva. 2015. The concept and impact analysis of a flexible mobility on demand system. *Transportation Research Part C: Emerging Technologies* 56 (2015), 373–392.

[7] R. Bhoraskar, N. Vankadhara, B. Raman, and P. Kulkarni. 2012. Wolverine: Traffic and road condition estimation using smartphone sensors. In *4th IEEE COMSNETS.* 1–6. https://doi.org/10.1109/COMSNETS.2012.6151382

[8] James Biagioni, Tomas Gerlich, Timothy Merrifield, and Jakob Eriksson. 2011. EasyTracker: Automatic Transit Tracking, Mapping, and Arrival Time Prediction Using Smartphones. In *9th ACM Sensys.* 68–81. http://doi.acm.org/10.1145/2070942.2070950

[9] Ionut Constandache, R Roy Choudhury, and Injong Rhee. 2010. Compacc: Using mobile phone compasses and accelerometers for localization. In *IEEE INFOCOM.* 1–9.

[10] Zhan Guo. 2011. Mind the map! The impact of transit maps on path choice in public transit. *Transportation Research Part A: Policy and Practice* 45, 7 (2011), 625–639.

[11] Jun Han, Emmanuel Owusu, Le T Nguyen, Adrian Perrig, and Joy Zhang. 2012. Accomplice: Location inference using accelerometers on smartphones. In *4th IEEE COMSNETS.* 1–9.

[12] Bo-Jhang Ho, Paul Martin, Prashanth Swaminathan, and Mani Srivastava. 2015. From Pressure to Path: Barometer-based Vehicle Tracking. In *2nd ACM BuildSys.* 65–74.

[13] Prashanth Mohan, Venkata N Padmanabhan, and Ramachandran Ramjee. 2008. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *6th ACM SenSys.* 323–336.

[14] Chris Nash. 2005. Privatization in transport. *Handbooks in Transport* 6 (2005).

[15] Sarfraz Nawaz and Cecilia Mascolo. 2014. Mining users' significant driving routes with low-power sensors. In *12th ACM SenSys.* 236–250.

[16] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *17th ACM SIGSPATIAL.* 336–343.

[17] Number of Buses Owned by the Public and Private Sectors in India 2015. (2015). https://data.gov.in/catalog/number-buses-owned-public-and-private-sectors-india.

[18] Magnus Nygård et al. 1999. A method for analysing traffic safety with help of speed profiles. (1999).

[19] Anson F Stewart. 2017. Mapping transit accessibility: Possibilities for public participation. *Transportation Research Part A: Policy and Practice* (2017).

[20] Yu-chin Tai, Cheng-wei Chan, and Jane Yung-jen Hsu. 2010. Automatic road anomaly detection using smart mobile device. In *conference on technologies and applications of artificial intelligence, Hsinchu, Taiwan.*

[21] Rohit Verma, Surjya Ghosh, , Niloy Ganguly, Bivas Mitra, and Sandip Chakraborty. 2017. Smartphone Sensor Data of Indian Cities for Public Bus. (2017). https://doi.org/10.17632/92yrxtv5gn.1

[22] Rohit Verma, Surjya Ghosh, Aviral Shrivastava, Niloy Ganguly, Bivas Mitra, and Sandip Chakraborty. 2016. Unsupervised annotated city traffic map generation. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.* ACM, 59.

[23] Rohit Verma, Aviral Shrivastava, Sandip Chakraborty, and Bivas Mitra. 2016. Margdarshak: A Mobile Data Analytics based Commute Time Estimator cum Route Recommender. In *Proceedings of the 3rd International on Workshop on Physical Analytics.* ACM, 31–36.

[24] Rohit Verma, Aviral Shrivastava, Bivas Mitra, Sujoy Saha, Niloy Ganguly, Subrata Nandi, and Sandip Chakraborty. 2016. UrbanEye: An Outdoor Localization System for Public Transport. In *Proceedings of the 35th IEEE INFOCOM.*

[25] T Vincenty. 1965. Transformation of Co-ordinates between Geodetic Systems. *Survey Review* 18, 137 (1965), 128–133.

[26] Michail Vlachos, George Kollios, and Dimitrios Gunopulos. 2002. Discovering similar multidimensional trajectories. In *18th IEEE ICDE.* 673–684.

[27] Christopher Willoughby. 2013. How much can public private partnership really do for urban transport in developing countries? *Research in Transportation Economics* 40, 1 (2013), 34–55.

[28] Hongzi Zhu, Yanmin Zhu, Minglu Li, and Lionel M Ni. 2008. HERO: online real-time vehicle tracking in Shanghai. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE.* IEEE, 942–950.