Processing and Management of GPS Based Bus Travel Time Data

Summer Research Fellowship Programme'18,
Indian Academy of Sciences

Akhila G P
SRFP Registration Number : ENGS438.

Department of Computer Science and Engineering, College Of Engineering, Guindy,
Anna University, Chennai-600025.

Guided By

Dr Lelitha Devi Vanajakshi
Associate Professor, Transportation Division.

Department of Civil Engineering, Indin Institute of Technology,Madras, Chennai-600036.

Contents

List of figures

List of Tables

Abstract

Transportation has become an essential component of life. Public Transport is one of the most sought after modes of transportation in the day-to-day life. In crowded metropolitan cities like Chennai, the reliability of MTC buses is greatly affected by factors like traffic congestion and route diversion and alteration, thereby hindering the free flow of traffic leading to huge delay. Location-enabled devices like Global Positioning Systems (GPS) can give real-time information from the vehicles, which can help in addressing the problem to a great extent. However, processing such data, that too in huge amount, requires proper data management strategies. The current study concentrates on this problem and develops programs to manage the raw data coming from GPS and process and store the data in ready to use format. The raw GPS data consists of position information (latitude and longitude) of the bus at uniform intervals. In this study, the data was reported every ten seconds. The data gets stored as a CSV file for each device separately for the whole day. To start with, the saved data was checked for data quality, which involved removal of error-prone (for instance, that lie outside the bounding circle) and incomplete data. In the next stage, the entire file was separated into multiple to and fro trips between the origin and destination, in the scheduled route. The segmented trips were stored in separate CSV files for each device. The processed data can be used to understand the bus travel time characteristics. The time-space plots were analyzed for this. The observations were made regarding the travel time of the bus.

Keywords : Data cleaning and structuring.

Abbreviations

1	GPS	Global Positioning Systems
2	MTC	Metropolitan Transport Corporation
3	WGS84	World Geodetic System 1984
4	CSV	Comma Separated Values

1 Introduction

Public transport is one of the most sought after modes of transportation. In densely populated and fast developing cities, the reliability of public transport is highly affected. Various factors like route diversion, intersections and traffic congestions cause the commuters huge delay. By predicting the arrival time of buses accurately, the delay in travel time can be reduced to a greater extent. This study is the first and foremost step required for the further research that aims at improving the travel time prediction. The project concentrates on creating a quality database by developing programs that manage and process the raw data. It ensures proper cleaning, structuring and processing of GPS data collected using GPS chips in buses.

Many researchers opine that the transport system is one of the important factors responsible for a nation's progress and economic growth. Public transportation reduces the traffic and is cost efficient as well. However, irregularities in scheduling greatly reduces the reliability of public transportation system. A greater accuracy or reliability can be achieved by collecting, processing and analyzing the position information. Then, the necessary details regarding the arrival/departure time of the bus, its real time location and accident/breakdown detection can be identified and interpreted. The analysis of real time information would improve the dependability of public transportation (Ghosh et al., 2017). A number of factors affect the accuracy of GPS measurements and lead to noises in GPS data. Therefore, the noise in the GPS must be filtered out to achieve accurate results. The anomalies have to be removed to achieve higher accuracy in predictions (Xu et al., 2016). Based on the literature review, the objectives of the study are,

- To perform automated quality check on the raw data.
- To check if the bus travels in its defined route.
- To identify and separate the to and fro trips completed by the bus in a particular day.
- To handle cut-services and segregate them from the normal trips.

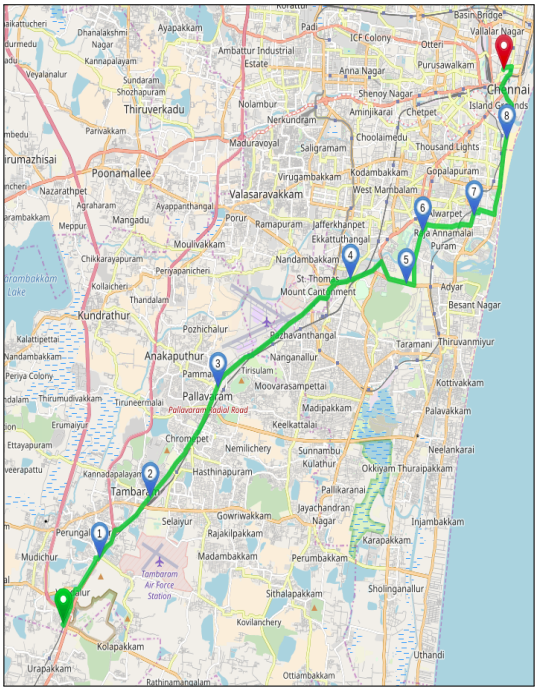
2 Methodology

2.1 Data collection

The geographic area chosen for the present study was Chennai, TamilNadu, India. The MTC buses running in Chennai were fitted with Global Positioning System chips to collect the raw data. The data containing the real time information about the position of the bus was reported every 10 seconds and was stored in CSV files for each device separately. The 21G bus route in Chennai is taken as a sample for testing the developed modules of code. The 21G bus runs across the Chennai city and also covers the fast growing sub-urban areas like Tambaram and Vandalur. The buses in the 21G route cover a length of 38 kilometers in a single run between the source (Broadway) and the destination (Vandalur Zoo). Figure 1 shows the route in which the 21G buses run and the way-points in the bus route are listed in Table 1.

Table 2.1: The major waypoints in the 21G bus route

S.No	**Way-points/Check points**
1	Perungalathur
2	West Tambaram
3	Pallavaram
4	Guindy
5	Gandhi Mandapam
6	Kotturpuram
7	Mylapore
8	Parrys



Source: Grasshopper and OpenStreetMap

Figure 2.1: Route map of 21G bus

2.2 Structure of raw data

The raw data in the .csv format is imported into the working environment as a list of dictionaries. Table 2 denotes the parameters of the raw data to be fed into the module for cleaning and structuring. The raw data files are named after the devices unique identification parameter IMEI number.

Table 2.2: Parameters of the raw data.

Paramter	**Units**
Position	Latitude and Longitude
Speed	Kilometers per hour
DeviceTime	Seconds
ServerTime	Seconds

2.3 Data cleaning

The initial step in the analysis is to remove the null values from the raw data. Only if both the keys (Latitude and Longitude) are non-empty, the entries from the raw data are appended to the cleaned data. The empty fields can be caused by various reasons like power failure and errors in transmitting or receiving. The next step converts the position parameter in the raw data to the standard degrees format(DD.D).The compass direction in the raw data is converted into signs. Since, the place of study (Chennai, Tamilnadu, India) lies in the northern hemisphere and longitudinally, in the eastern hemisphere, the position parameter is always positive. The code for doing this is shown in Listing 1.

```
if((row['Latitude']!='' and  row['Longitude']!='')) :
                row['Latitude']=re.sub("[^\d.]","",row['Latitude'])
                row['Longitude']=re.sub("[^\d.]","",row['Longitude'
    ])
```

Listing 1: Employing a regular expression(regex)

Here, row corresponds to each entry in the raw data. The alphabetic part, representing the compass direction is removed from the position parameter by employing a regex (regular expression).Table 3 shows a sample of the position parameters- latitude and longitude, after the execution of the above snippet.

Table 2.3: Output obtained after employing the regex.

Input format	Output format
13.08764N,80.28357E	13.08764,80.28357
13.08759N,80.2836E	13.08759,80.2836
13.08721N,80.28364E	13.08721,80.28364

The entries in the cleaned data are sorted based on the device time using the built-in function sort() as shown in Listing 2.

```
gps_data.sort(key=itemgetter(' DeviceTime'))
bounding_box=[(12.768441444,79.9262830742)
,(13.28758473421,80.3243304968)]
for i in  range(0,len(gps_data)):
    if(bounding_box[0][0] <= float(gps_data[i]['Latitude']) and
 float(gps_data[i]['Latitude']) <= bounding_box[1][0] and
bounding_box[0][1] <= float(gps_data[i]['Longitude']) and float
(gps_data[i]['Longitude']) <= bounding_box[1][1]):
        gps.append(gps_data[i])
```

Listing 2: Identifying the gps points that lie outside the considered boundary

range.

The variable `gps`, in Listing 2, contains only the entries from the input data that lie within a particular range. The list variable `bounding_box` [5] (Source: Bounding Box) defines a boundary region for the place of study and ensures that the data to be transferred to the processing stage does not contain inconsistent and irrelevant information.

After performing the above three steps of quality check on the raw data, an appreciable amount of error-prone data was removed from each file.

2.4 Trip segmentation

After cleaning the data, the source and destination depots are identified and the data is segmented into a number of to and fro trips. A trip is the movement of the bus from the source to the destination in a proper route.

In order to ensure that the bus travels in its destined route, a number of way-points in the route are taken into consideration. A trip is valid and will be stored only if the bus crosses all the check points on its way. The sample check points taken into consideration for the 21G route are mentioned in Listing 3.

```
#this is the depot point for origin
src_latlng=(13.0875,80.2839)

#this is the depot point for destination
dest_latlng=(12.8804,80.0804)

#add any number of check points or waypoints in between
check1_latlng=(13.03378,80.26794)
check2_latlng=(13.0263702,80.264001)
check3_latlng=(13.00822,80.2099)
check4_latlng=(12.92761,80.11883)
```

Listing 3: A list of source,destination and check-points in the route

The `geographiclib` package from python is used to compare two gps co-ordinates. The Geodesic module from the package has an in-built function `Inverse()` which takes two sets of lat-lng as input and returns a dictionary. The example is demonstrated in Listing 4.

```
>>> from geographiclib.geodesic import Geodesic
>>> geod=Geodesic.WGS84
>>> point1=(13.08633,80.28357)
>>> point2=(13.08589,80.28355)
>>> d=geod.Inverse(point1[0],point1[1],point2[0],point2[1])
>>> d
{'azi2': -177.4487490860769, 'lon2': 80.28355, 'lat2': 13.08589, '
    s12': 48.72603413268235, 'azi1': -177.44874455777324, 'a12':
    0.0004391105529036961, 'lon1': 80.28357, 'lat1': 13.08633}
>>> type(d)
<class 'dict'>
```

Listing 4: Calculating the geodesic distance between two gps points.

The value of `s12` key is the shortest distance between two GPS coordinates on the earth surface in meters. The `Inverse()` function from the Geodesic module uses the standard WGS datum to perform calculations. The various check-points (variables) in the bus route are made true whenever the distance between the check-point and the entry in the data is less than 100 meters.

The trips identified from the data are stored in separate CSV files named after the devices IMEI number and the starting time of the trip.

2.5 Handling cut services

A cut service is a trip undertaken by the bus between any two major bus stops in the route rather than connecting the source and the destination. The cut service trips can be tracked from the data file by processing the data that was discarded after handling the normal to and return trips between the origin and destination. The possible cut service depots and the routes of such services are identified and structured into a list.

```
cut_service_route1=[src_latlng,check2_latlng,check3_latlng,
    check4_latlng]
cut_service_route2=[src_latlng,check2_latlng,check3_latlng]
cut_service_route3=[check2_latlng,check3_latlng,check4_latlng]

#the whole list of possible cut-services
cut_ser_list=[cut_service_route1,cut_service_route2,
    cut_service_route3]
```

Listing 5: Defining the route for each cut service and the list of possible cut-services

where the `src_latlng`, `check2_latlng`, `check3_latlng`, `check4_latlng` are initialized in Listing 3. The trip being tracked, is checked to see,

if it matches with any of the routes in the `cut_ser_list`. The trips that satisfy the condition are accepted and are stored in a separate CSV file.

The files are named after the devices IMEI number, the cut-service route identified and starting time of the trip.

3 Results and Conclusion

The codes were tested on sample data files and Table 4 indicates the properties of the sample files.

Table 3.1: Summary of sample files taken for study.

Route taken for study	**Date taken for study**	**Number of devices taken for study**
21G route, Chennai	2018-05-07 (7th May, 2018)	17 devices

The module `clean_data()` (Source code in appendix) discarded an appreciable amount of error entries from the raw data and the number of entries discarded corresponding to each device is listed in Table 5.

Table 3.2: Number of entries discarded from each device.

Device ID	**Original Length of the file**	**Length after cleaning**
Device 1	5942	5940
Device 2	7099	6974
Device 3	6367	6088
Device 4	5825	5557
Device 5	7704	7688
Device 6	5278	5232
Device 7	1853	1853
Device 8	3321	3319
Device 9	5451	2738
Device 10	7054	7054
Device 11	4065	2338
Device 12	5617	5617
Device 13	3887	3873
Device 14	7457	7204
Device 15	147	146
Device 16	7129	6699
Device 17	5490	5481

The module `trip_segmenter()` was executed and the onward and return trips were identified from 7 devices out of 17 devices. Table 6 summarizes the results of this part of the code. The input files were checked manually to find out the number of trips. The results matched with the output obtained by executing the code. Onward trips refer to the trips between the source (Broadway) and destination (Vandalur Zoo) and return trips refer to the trips between the destination (Vandalur Zoo) and the source (Broadway).

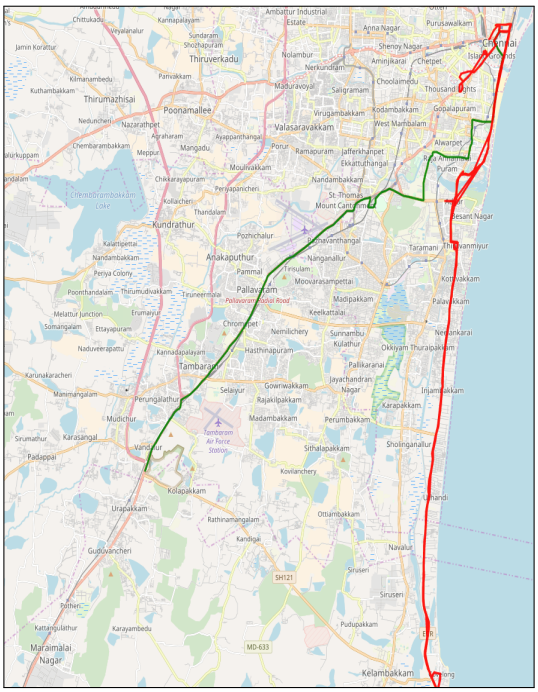
Table 3.3: Summary of test run on sample data.

Device ID	**Number of onward trips**	**Number of return trips**
Device 1	2	2
Device 2	4	3
Device 3	3	3
Device 10	3	4
Device 12	4	3
Device 16	2	1
Device 17	3	3

The following observations were made from the data files that does not contain any onward or return trips in the route taken for study.

- The buses with the device (GPS chip) implanted in it, took another route instead of the route taken for study (21G).
- The buses with the device implanted in it, were retained in a specific depot, for reasons like regular maintenance, repair or break down.

Figure 2 expresses a sample visualization, where the red trajectory corresponds to the movement of the bus in another route instead of 21G bus route and the green trajectory corresponds to the movement of the bus in the correct route.



Source: GPS Visualizer and OpenStreetMap

Figure 3.1: Visualization of accepted and rejected trajectory.

The module `cut_service()` handles the cut service trips and the execution of the code identified eight cut-service trips (from three possible cut-service routes) in the studied route. The various cut-service trips tracked from each device are listed in Table 7. The manual checking of the input files resulted in the same number of cut-services as identified by the code.

Table 3.4: Summary of the cut service trips tracked.

Device ID	**cut_service_ route1**	**cut_service_ route2**	**cut_service_ route3**
Device 1	0	1	0
Device 3	2	0	0
Device 10	1	0	0
Device 12	0	0	1
Device 16	1	0	0

The value for the columns `cut_service_route1`, `cut_service_route2`, `cut_service_route3` were initialized in Listing 5.

3.1 Time-Space Plots

Once the onward and return journeys were identified, to understand the trip characteristics, time-space based trajectory plots were made. The onward and return trips are plotted with time in x-axis and distance (in meters) in y-axis. 19 trips with different starting time between the source and destination are plotted and the output is shown in Figure 3. Trips in the opposite direction, between destination and source, are plotted in Figure 4. The plots are obtained from the pythons `matplotlib` package and `pyplot` module. The x-axis is formatted to contain datetime ticks at regular intervals (one hour) in these graphs.

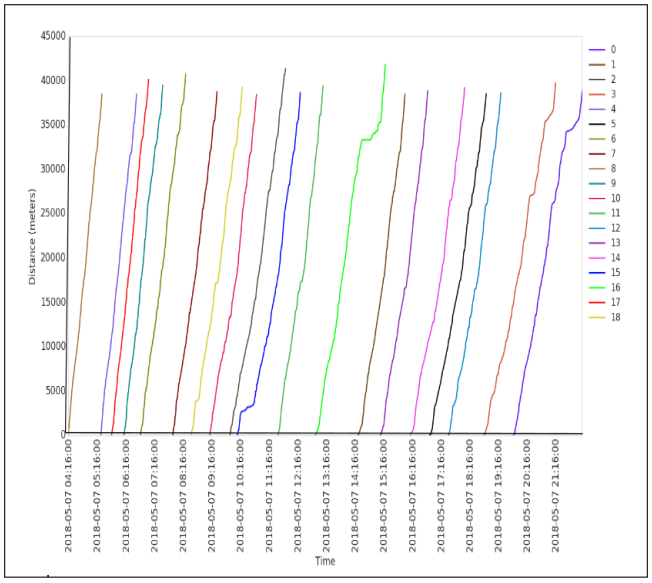


Figure 3.2: Visualization of trips between source and destination

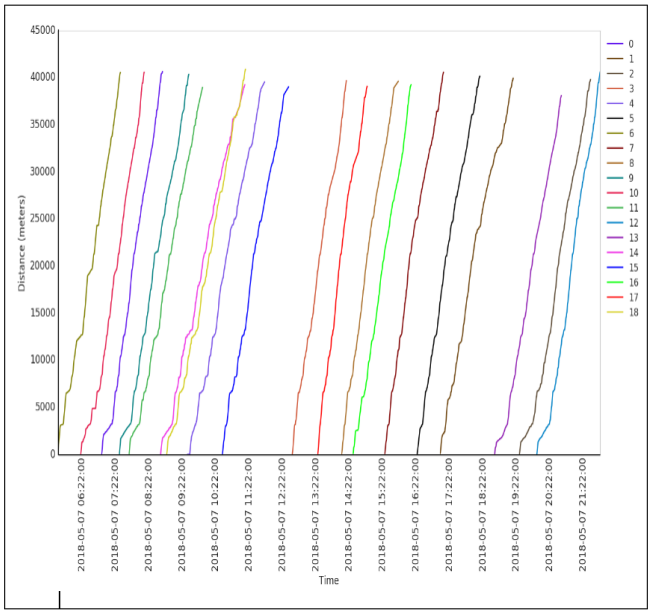


Figure 3.3: Visualization of trips between destination and source.

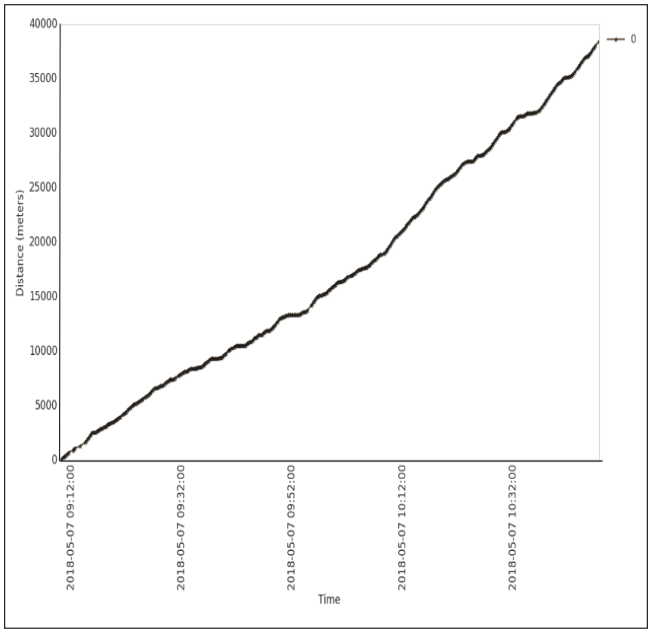


Figure 3.4: Plotting a sample trip between the origin and destination.

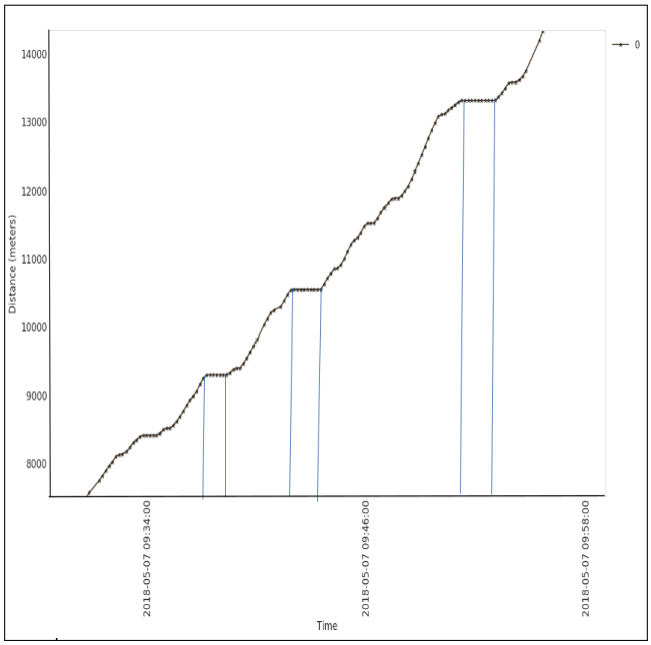


Figure 3.5: A close up look at a portion of the trip containing dwell time.

Figure 5 represents a sample time vs distance plot between the source and the destination. In this sample plot, the trip has started from the source/initial position at 09:09 am and has ended at 10:47 am. From the graph, it can be seen that the trip covered a distance of 38679 meters. From this plot, one can see the variation in speeds and delay points during the journey. The plot is zoomed to check for dwell time at a particular bus stop and is shown in Figure 6. The horizontal line indicates the bus has no movement and is stationary. The vertical lines drawn in the plot indicates the time range at which the bus has no movement.

References

1. Raksha Ghosh,R Pragathi, S Ullas, Surekha Borra (2017), *Intelligent transportation systems: A survey*, In: International Conference on Circuits, Controls, and Communications (CCUBE), Dec 2017.

2. Zhenzhen Xu, Mingfei Jia, Lu Li, Shuo Yu, Jiancheng Yu, and Shijie Liu (2016), *Data Preprocessing and Fitting Algorithm Based on Marine Data Sampled by Multiple Underwater Gliders*, In: 12th World Congress on Intelligent Control and Automation (WCICA), Guilin, China.