Multivariate Data Analysis

Project

Analyzing GPS data history of vehicles, in an urban journey.

Submitted by

Shara Rhagha Wardhan B (ME11B121)

Sai Prabhakar P A (ME11b053)

Content:

Abstract:

GPS or Global Positioning System has become an integral part of our daily life. However, the processing and analysis of GPS data remains in important issue.

Different kind of GPS Analysis can be done. They are as enumerated below:

Terrain Analysis:
Terrain analysis is nowadays considered to be an important analytical tool for Geographical and Geological studies. GPS in this regard plays an important role in performing the terrain analysis. GPS, as we know, can able to generate X coordinate, Y coordinate and Z coordinate i.e. latitude, longitude and altitude above the mean sea level respectively. They give a virtual 3-D effect to the topography and helps in studies and further research.

Area Measurement:
Area Measurement through GPS is widely done. In this case, the GPS receivers are made to record the coordinates of various points throughout the perimeter of the concerned area. Later, these data are processed and calculation is done to compute the area measurement of drainage basin, city, town, and market are done through GPS.

Measurement of Path Length:
GPS technologies are used in local transportation and communication network mostly in European nations. GPS are employed in measurement of path length between various places. It also enables the passengers to visualize their vehicle & their positions for which they are waiting in a particular station.

Here we use GPS data for finding the path length and waiting time. Of a vehicle which **can be used in taxis and other public transport** to make the **service secure and fair for the customers**, by monitoring the path of the vehicle, and charging based on the travel and waiting time.

Training data generation:

Training data was generated from **within the campus (IITM), using the institute bus**. GPS data was collected using **GPS Tracker Service** software developed by **Gyandata**.

Instantaneous velocity input was recorded from the **odometer of the bus**.

Attempted Algorithm

Distance calculation:

Kernel PCA:

We attempted to denoise z exploiting the non-linear relation $x^2 + y^2 + z^2 = R^2$. But this method fails as it is an Ordinary Least Squares solution in z and not a Total least squares solution

Spline fitting:

This is not a good technique as there is no notion of denoising and the prediction of data in the unknown time region is very bad

Velocity calculation:

Method one:

Fitting $x^2\ vs\ time,\quad y^2\ vs\ time,\ Z^2\ vs\ time$ separately and estimating $\hat{x}, \hat{y}\ and\ \hat{z}$, for the time instance in the data point then finding the velocity for time(t) using

$$v(t) = \frac{\sqrt{\left(\hat{x}(t_1) - \hat{x}(t_2)\right)^2 + \left(\hat{y}(t_1) - \hat{y}(t_2)\right)^2 + \left(\hat{z}(t_1) - \hat{z}(t_2)\right)^2}}{t_2 - t_1}$$

Where t_2 , $t_1\ are$ the time instance in the data covering t

Method two:

We attempted to fit $x^2\ vs\ time,\quad y^2\ vs\ time,\ Z^2\ vs\ time$ separately. Then finding the x, y and z for every second using

$$v(t) = \frac{\sqrt{\left(\hat{x}(t_1) - \hat{x}(t_2)\right)^2 + \left(\hat{y}(t_1) - \hat{y}(t_2)\right)^2 + \left(\hat{z}(t_1) - \hat{z}(t_2)\right)^2}}{t_2 - t_1}$$

Where t_2 , $t_1\ are$ the time instance in the data covering t and $\quad t_2 = t_1 + 1.$

Method three: Finding cumulative distance travelled, by adding distance travelled in each time instance in data point, using

$$distance(t_1) = distance(t_2) + \sqrt{\left(\hat{x}(t_1) - \hat{x}(t_2)\right)^2 + \left(\hat{y}(t_1) - \hat{y}(t_2)\right)^2 + \left(\hat{z}(t_1) - \hat{z}(t_2)\right)^2}$$

Fitting the least square spline to distance vs time and then finding velocity either by finding

$$v = \frac{\hat{d}(t_2) + \hat{d}(t_1)}{t_2 - t_1}, where\ t_2\ and\ t_1\ belong\ to\ time\ array$$

OR USING

$$v = \frac{\hat{d}(t_2) + \hat{d}(t_1)}{t_2 - t_1}, where\ t_2 = t_1 + 1\ .$$

Waiting time:

Velocity based:

After finding the velocity at all-time instances, we set a threshold velocity, assuming the vehicle if moving have to travel with velocity more than the threshold other-wise it is stationary.

$$travell\_history = \begin{cases} v(t) > threshold, & travelling \\ v(t) < threshold, & waiting \end{cases}$$

Mean velocity based:

Finding the $mean(velocity)$.

Then finding the travel velocity time,

$$time_{travel} = \frac{distance_{total}}{mean(velocity)}$$

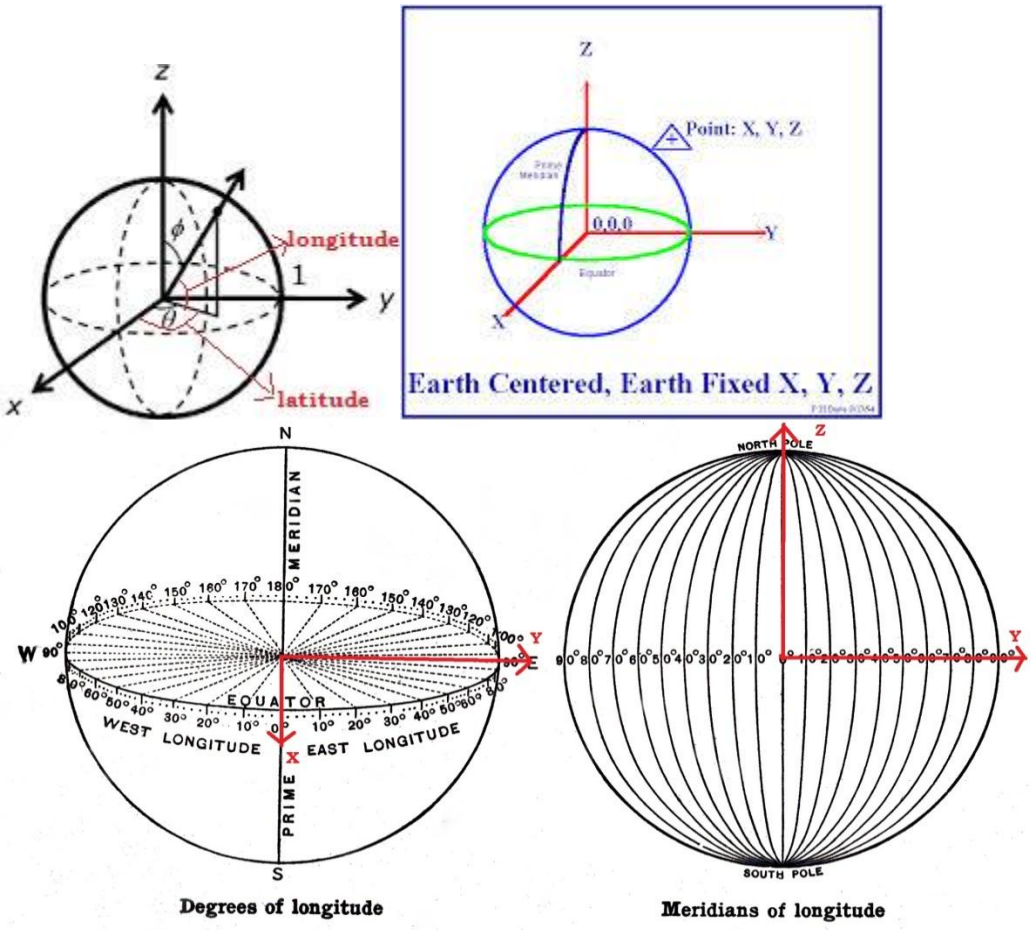
Then finding

$$time_{waiting} = time_{total} - time_{travel}$$

Final Algorithm (Best)

Step 1: Transformation of coordinates

Conversion of (radius of earth, latitude, longitude) to spherical coordinates and then to Cartesian coordinates



From the figures, the coordinate transformation is quite clear

$\theta = latitude$	$\phi = 90° - longitude$	
$x = Rcos\theta sin\phi$	$y = Rsin\theta sin\phi$	$z = Rcos\phi$

Step 2: Use PCA to denoise the data

- We know that there exists a linear relationship between x^2 , y^2 and z^2 as $x^2 + y^2 + z^2 = R^2$ where R is the radius of the Earth.
- So we use PCA on the x^2 , y^2 and z^2 shifted by mean (as we expect an offset term) and scaled by accuracy (since 68% confidence implies that the confidence interval (or accuracy) = standard deviation) with the heuristic that 99.99% of the variance should be captured.

Step 3: Use functional PCA on the denoised data to further denoise the data

- Again exploiting the same linear relation between x^2 , y^2 and z^2, we use functional PCA to the previously estimated data.
- We fit x^2 , y^2 and z^2 as a function of time using least square spline basis since nothing can be predicted about the path of travel and get the estimated values of x, y and z.
- We run an optimization code to minimize the error and at the same time to minimize the order of polynomial and the number of knots chosen.
- We iterate the order of polynomial between 3 and 8 (in MATLAB code cubic spline is minimum required and for this the argument is 4 since there are 4 coefficients to be determined) and the number of knots between 3 and 80.

We try to minimize the sum

$$Obj = \log(error) + (number\ of\ parameters + 1) * \frac{2}{N}$$

$$error = \sum_i \left(\frac{xhat2_i - xhat_i}{accuracy_i}\right)^2 + \left(\frac{yhat2_i - yhat_i}{accuracy_i}\right)^2 + \left(\frac{zhat2_i - zhat_i}{accuracy_i}\right)^2$$

$xhat_i, yhat_i$ and $zhat_i$ $are\ the\ values\ of\ x, y$ and z $after\ getting\ denoised\ by\ PCA$

$xhat2_i, yhat2_i$ and $zhat2_i$ $are\ the\ values\ of\ x, y$ and z $after\ getting\ denoised\ by\ functional\ PCA$

Number of parameters determined is got from the structure stored in MATLAB when the fit is made

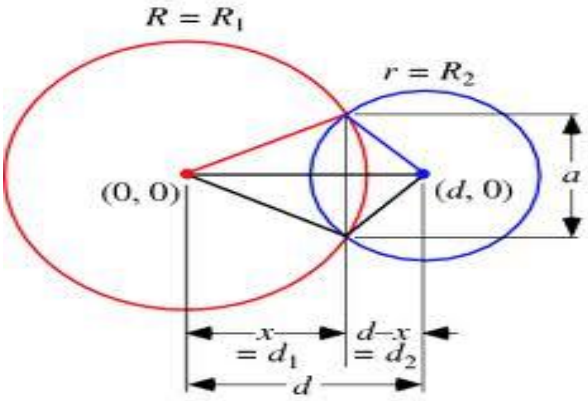
Step 4: Calculate the distance and velocity using the denoised data

- We find the optimal solution and use the optimal order and optimal number of knots to project the data onto the least square spline and get the denoised values of x, y and z.
- Using the denoised data we find the total distance and we compute the error in the total distance calculation. We compute the velocity at that instant by using $velocity = \frac{distance}{time}$.

Step 5: Compute the standing time using the denoised distance and velocity and accuracy

- We compute the standing distance by
  - Associating a probability to the intersection region of the accuracies and
  - Giving a threshold to the computed velocity.
- This gives a good estimate of the standing time
- The threshold is chosen such that it takes into account some parameters of the given dataset so that the parameter varies from dataset to dataset

Case 1: The distance between two consecutive points is less than the sum of the accuracy of the two points. In this case we get an estimate of the standing time



$$IA = r^2 \cos^{-1}\left(\frac{d^2 + r^2 - R^2}{2dr}\right) + R^2 \cos^{-1}\left(\frac{d^2 + R^2 - r^2}{2dR}\right) - \frac{1}{2}\sqrt{(-d + r + R)(d + r - R)(d - r + R)(d + r + R)}$$

Where IA indicates the intersection area

$$Area1 = \pi R^2 \qquad Area2 = \pi\, r^2$$

$$Probability\ that\ the\ 1st\ point\ lies\ in\ the\ intersection\ area\ = \frac{IA}{Area1} \times 0.68$$

$$Probability\ that\ the\ 2nd\ point\ lies\ in\ the\ intersection\ area\ = \frac{IA}{Area2} \times 0.68$$

Where 0.68 is the probability that the point is found the circle of the drawn radius

So the estimate of the standing time between the two points $= t(i) \times \frac{IA^2}{Area1 \times Area2} \times 0.68^2$

Where t(i) is the time difference between the two points.

- When the second circle lies entirely inside the first circle completely, $IA = Area2$
- When the first circle lies entirely inside the second circle completely, $IA = Area1$

Case 2: When the velocity is less than a threshold

$$average\ velocity = \frac{total\ distance}{total\ time}$$

$$threshold = \frac{average\ velocity}{standard\ deviation\ of\ velocity}$$

- If velocity is greater than threshold, then add the time between the points to the standing time.
- This threshold is just a heuristic which seems to work reasonably well.

Step 6: Finding the missing data using functional PCA for timestamp with interval of one second

- Create a new timestamp with interval of one second from 0 seconds to the last observed time (in seconds) and get the functional value of x^2 , y^2 and z^2 and get the values of x, y and z from that.

Step 7: Get the velocities for every time instant and compare with velocities given by the user

- Procedure for calculation of velocity is as follows

  Let $a = x^2$, b = y^2 and $c = z^2$

$$\frac{da}{dt} = 2x\frac{dx}{dt}, \qquad \frac{db}{dt} = 2y\frac{dy}{dt}, \qquad \frac{dc}{dt} = 2z\frac{dz}{dt}$$

In this from step 6, $dt = 1$ implying

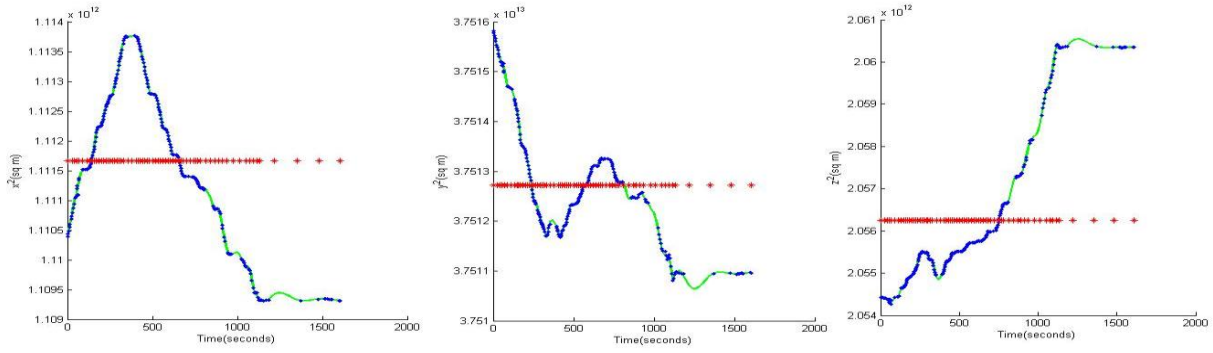
$$dx_i = \frac{a_i - a_{i-1}}{2\,x_i}\ , dy_i = \frac{b_i - b_{i-1}}{2\,y_i}\ , dz_i = \frac{c_i - c_{i-1}}{2\,z_i}$$

$$v_i = \sqrt{(dx_i)^2 + (dy_i)^2 + (dz_i)^2}$$

- Then we filter out the velocities for the given timestamps and display the error in the computed and observed velocities.

RESULTS (for our data)

Fitted data (x^2, y^2 and z^2 as a $function$ of $time$ $using$ $least$ $square$ $splines$)



The red dots indicates the knots chosen
The blue dots indicates the actual data points
The green line indicates the least square spline

Output Acquired:

Total distance = 5.3 km

Total distance computed = 5.3136 km

Error in distance = 0.013585 km

Waiting time = 505 seconds

Computed waiting time = 510.62 seconds

Error in waiting time = 5.6229 seconds

Comparison of the instantaneous speed

Time Stamp (seconds)	Experimental Speed (kmph)	Computed Speed (kmph)	Error in Speed (kmph)
168	50	41.30341	8.696593
244	30	26.71764	3.28236
270	40	18.64973	21.35027
322	50	25.98356	24.01644
844	60	20.89077	39.10923
915	40	34.77214	5.22786
940	20	22.89516	-2.89516
1020	50	29.16362	20.83638