
Probe Data Processing

— Jordan Zeeb —
Senthil Palanisamy

Probe matching

- Before we can estimate slopes from probe data, we must match the appropriate probes to each link.
- Many links consisted of multiple sublinks, so we first must first match the probes to the sublinks before we can average their slopes together, weighted by their length.
- By creating a simple distance tolerance around each probe, we were able to find all of its nearby sublinks, which we then used to pick the sublink most appropriate for the probe.
- Once all the probes are matched with sublinks, we find consecutive pairs of probes with the same sampleID, and calculate the slope relative to each other.
- If multiple pairs of probes are matched with a sublink, then their values are averaged together weighted with their distance from each other.

Slope Calculation

-In order to calculate the slope, we must first convert the lat and lon of each probe point to UTM coordinates. UTM cords have 60 zones, in addition to the “Easting” and “Northing” cords, so we also checked if all the links and probes were in the same zone in order to remove that information, which they were.

-With all of our data in meters, we can now calculate the slope by the equation:

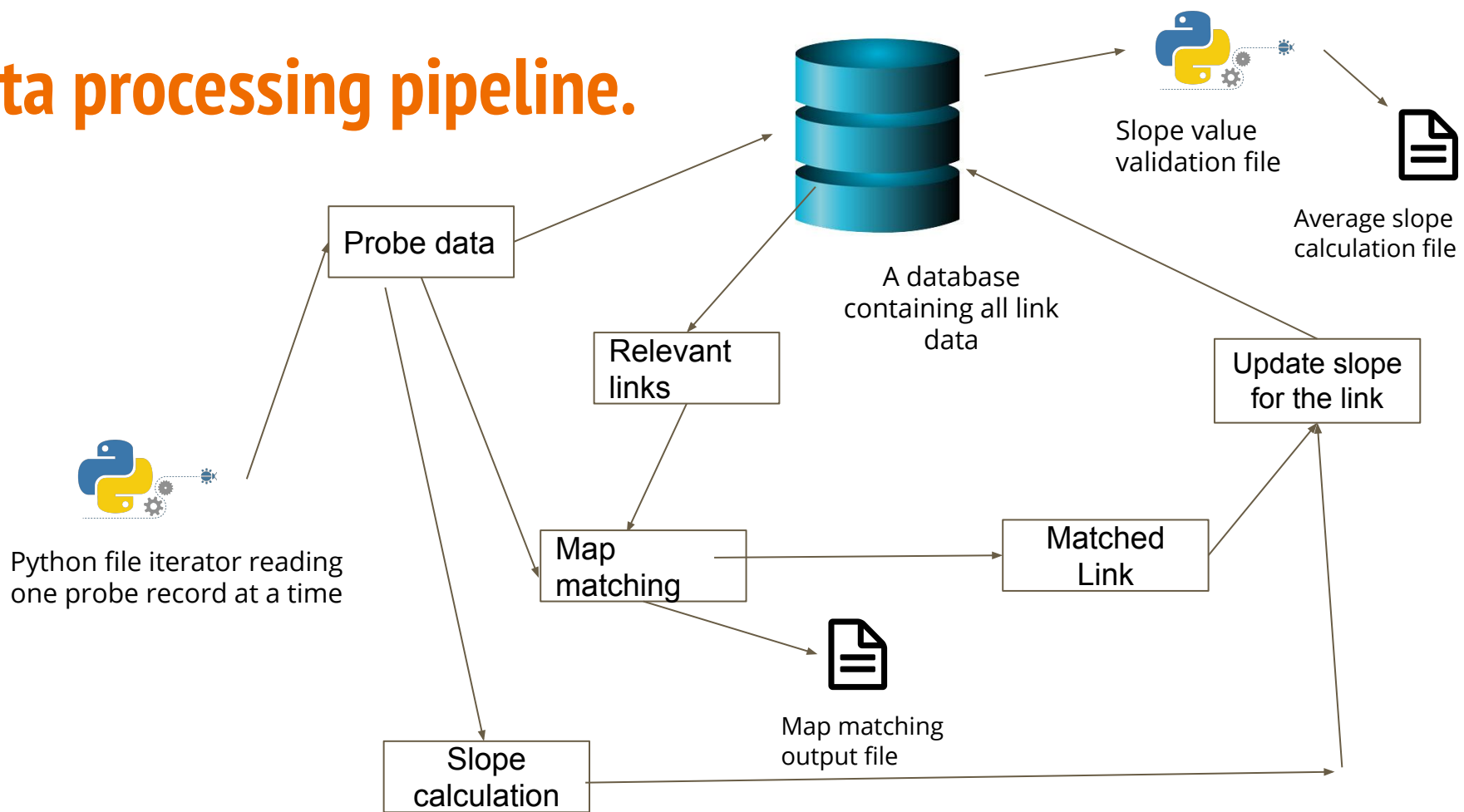
$$Slope_{probes} = \frac{(altitude_{probe2} - altitude_{probe1})}{\sqrt{((X_{probe2} - X_{probe1})^2 + (Y_{probe2} - Y_{probe1})^2)}}$$

Where X and Y are the UTM coordinates “Easting” and “Northing” respectively

Dataset explained

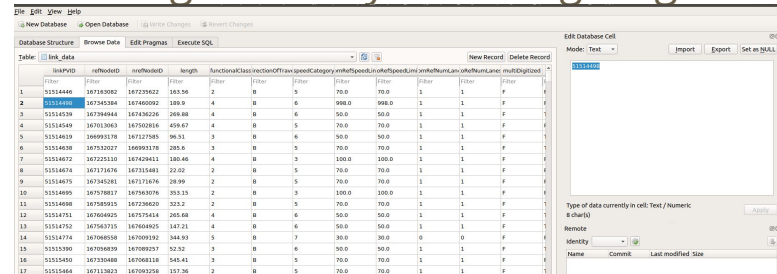
- The link dataset consisted of lots of relevant data, but we were only interested in the latitude, longitude, and altitude information which was all stored in the one column "shapeInfo"
- Most of the links do not contain altitude data, but for the ones that do we can calculate the slope in the same way we did for the probes.
- With this we have slope data for individual sub links which we can compare to our probe data.

Data processing pipeline.



Data processing pipeline explained

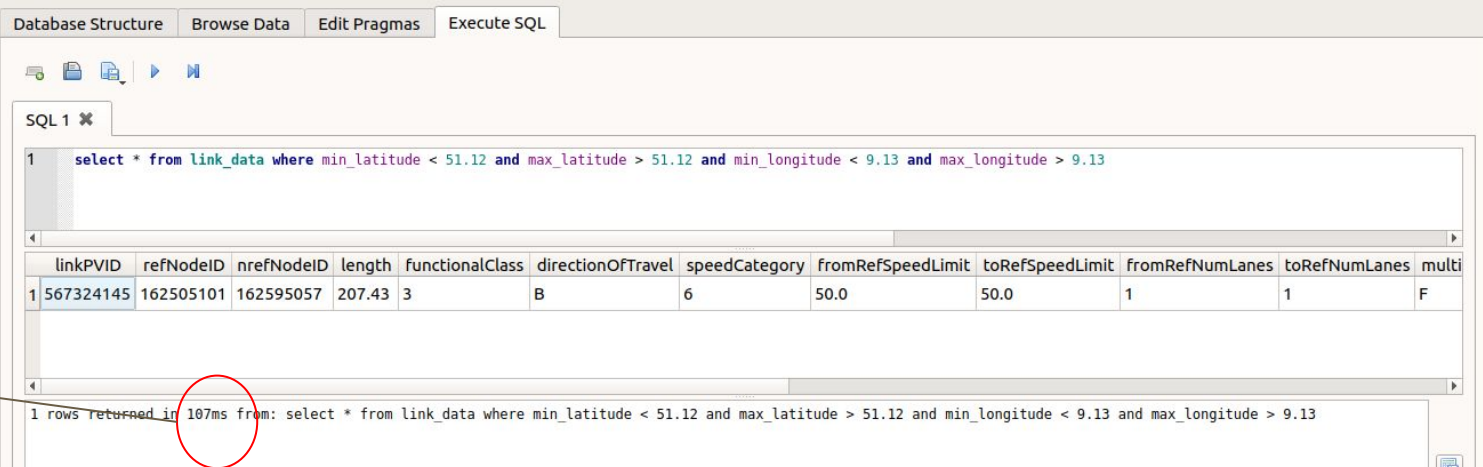
1. All link data is uploaded into a database.
2. For a given probe point, all neighbouring links are fetched from the DB (**This only takes a few ms!**)
3. Then among all the neighbouring links, a map matching function finds the best matching link for the given probe point (A point based map matching technique is used)
4. The slope of the probe point is calculated using the probe data.
5. The map matching data is written to an output file and the slope for each link is updated in the db
6. Finally a script calculates the average slope for each road link based on all the slope estimate and finds an accuracy of the whole algorithm by matching it against the ground truth data



linkID	refNodeID	length	speedLimit	category	inflow	outflow	inflow	outflow	inflow	outflow	inflow	outflow
1	151514446	167132082	183.36	2	R	5	70.0	1	1	F	1	F
2	151514410	1671343384	167460082	189.5	4	R	6	998.0	998.0	1	1	F
3	151514339	167390484	167436236	268.68	4	R	6	50.0	50.0	1	1	F
4	151514548	167111963	167502856	408.62	4	R	5	70.0	70.0	1	1	F
5	151514619	166993178	167127585	96.31	3	R	6	50.0	50.0	1	1	F
6	151514628	167332027	166993178	285.5	3	R	5	70.0	70.0	1	1	F
7	151514672	167221130	167429411	180.48	4	R	5	100.0	100.0	1	1	F
8	151514674	167171676	167333481	22.02	2	R	5	70.0	70.0	1	1	F
9	151514675	167345281	167171676	28.99	2	R	5	70.0	70.0	1	1	F
10	151514695	167170817	167563076	353.15	2	R	3	100.0	100.0	1	1	F
11	151514698	167508915	167336639	32.3	2	R	5	70.0	70.0	1	1	F
12	151514751	167608825	167575414	265.68	4	R	6	50.0	50.0	1	1	F
13	151514752	167563715	167608825	147.31	4	R	6	50.0	50.0	1	1	F
14	151514774	167608858	167609182	264.93	3	R	7	50.0	50.0	0	0	F
15	151515390	167058839	167088237	52.82	3	R	6	50.0	50.0	1	1	F
16	151515450	167330488	167088118	545.41	3	R	5	70.0	70.0	1	1	F
17	151515464	167111923	167088258	137.36	2	R	5	70.0	70.0	1	1	F

Advantage of using Databases

1. The biggest bottle neck of the whole pipeline is the fact that we have check the probe point against each map link. There are more than 2 lakh map points. If we simply did the brute force search with a file based operation, it would simply not be realistic.
2. On the other hand, we could use SQL to query a data from the DB in a few ms.
3. In order to setup such a mechanism of data access, We preprocessed all the shape files of road link and add 4 columns: min_latitude, max_latitude, min_longitude, max_longitude. These are the minimum and maximum across all shape points of the link. Some tolerance was added to the ranges so that a SQL query can be written (shown below) to fetch all road links that could potentially contain the given probe point.



The screenshot shows a database management interface with tabs for 'Database Structure', 'Browse Data', 'Edit Pragmas', and 'Execute SQL'. The 'Execute SQL' tab is active, displaying a SQL query in a text area. Below the query, a table of results is shown. The table has columns: linkPVID, refNodeID, nrefNodeID, length, functionalClass, directionOfTravel, speedCategory, fromRefSpeedLimit, toRefSpeedLimit, fromRefNumLanes, toRefNumLanes, and multi. The first row of data is highlighted. At the bottom, a status bar indicates '1 rows returned in 107ms from: select * from link_data where min_latitude < 51.12 and max_latitude > 51.12 and min_longitude < 9.13 and max_longitude > 9.13'. A red circle highlights the '107ms' value, and an arrow points from the text 'Databases allow filtering across 2 lakh data points in 107ms' to this circle.

SQL 1 ✕

```
1 select * from link_data where min_latitude < 51.12 and max_latitude > 51.12 and min_longitude < 9.13 and max_longitude > 9.13
```

linkPVID	refNodeID	nrefNodeID	length	functionalClass	directionOfTravel	speedCategory	fromRefSpeedLimit	toRefSpeedLimit	fromRefNumLanes	toRefNumLanes	multi
1 567324145	162505101	162595057	207.43	3	B	6	50.0	50.0	1	1	F

1 rows returned in 107ms from: select * from link_data where min_latitude < 51.12 and max_latitude > 51.12 and min_longitude < 9.13 and max_longitude > 9.13

Databases allow filtering across 2 lakh data points in 107ms

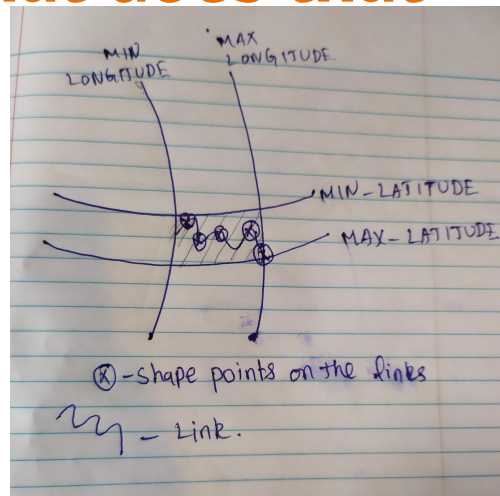
Latitude, longitude filtering in DB? What does that mean

1. These min latitude, max latitude, min longitude, max longitude together defines the smallest spherical sector that contains all the shape points of the road link.
2. Once stored in DB, these data columns will be extremely useful for fetching all relevant links from a db given a probe point.
3. More concretely, given a probe point latitude and probe point longitude, all the records we have pull out are the ones which satisfy the following relationship

Min latitude \leq Probe latitude \leq Max latitude

Min longitude \leq Probe longitude \leq Max longitude

4. It must be noted that these links are NOT the final map matched links. It is just a quick means of filtering all relevant links from a DB



Failed multiprocessing experiment to speed things further

1. It can be seen that each probe data processing is completely independent of other data processing
2. Hence, we set out to do multiprocessing so that multiple probe data records could be processed in parallel

BUT ..

We used SQLITE for Database management. It seems to be one of the most primitive databases and doesn't support concurrent query executions. Therefore, this experiment failed but it should be possible to parallelise each processing of probe data using Multiprocessing on more sophisticated databases.

Note:-

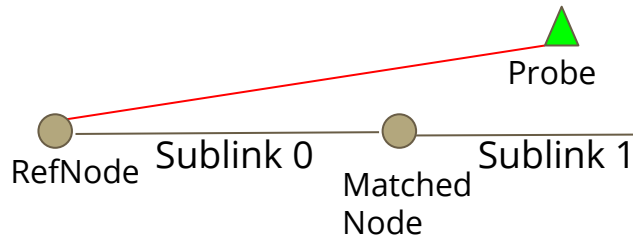
We also tried out strategies to have two processes: one for fetching data from database and the other for processing data. Since data processing time (a few ms) is much smaller than the data fetching time (100 ms), the timing is actually increased due to the time lost in process synchronisation

It must be noted that for multi threading does not give any speed in Python (All threads run on the same core due to GIL (Global interpreter Lock)). Hence, one should use multiprocessing packages for achieving parallelism

Method I Map matching- Each link is a sequence of control points

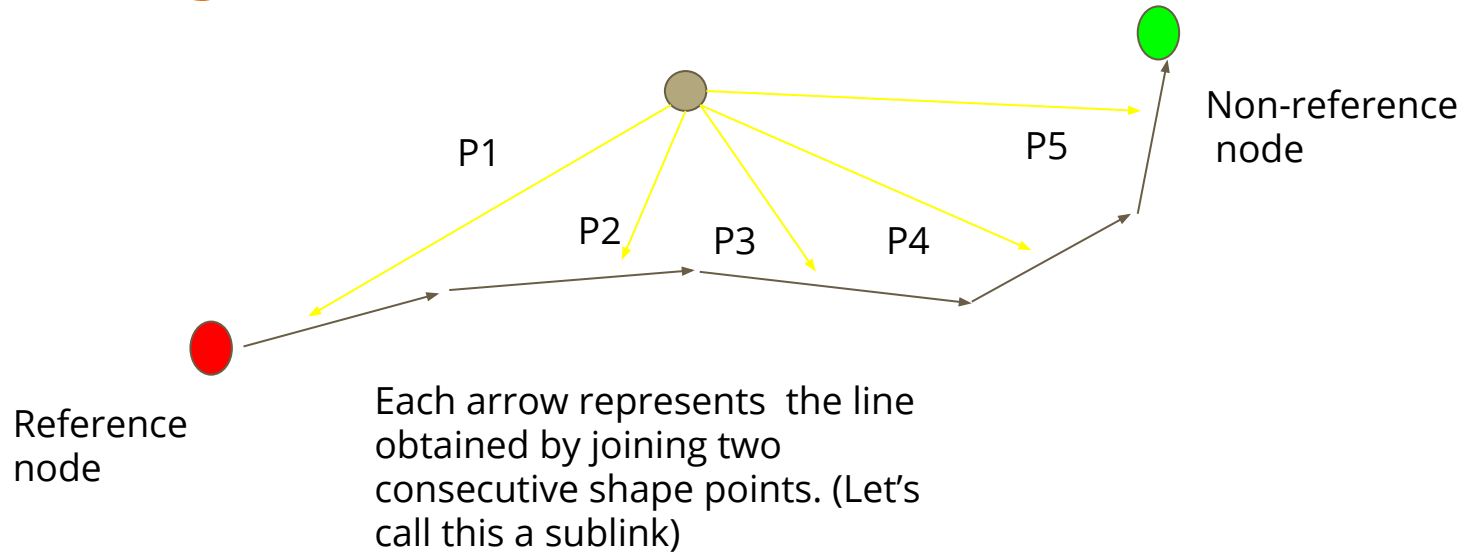
-After we find all of the link nodes that fall within each probes distance tolerance, we can match a link to each probe by simply finding the node with the shortest distance to the probe. This again, is done after a UTM conversion of all points and link nodes.

-After we find the matched node, we know the probe can only be in the sublink before, or after the node, so we compare the distance between the links refNode and the matched node with the distance between the refNode and the probe in order to determine which one it is. This can be seen below.



Since $\text{dist_ref_matched} < \text{dist_ref_probe}$, then Sublink 1 of this link is matched to the probe.

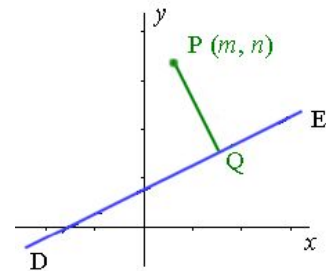
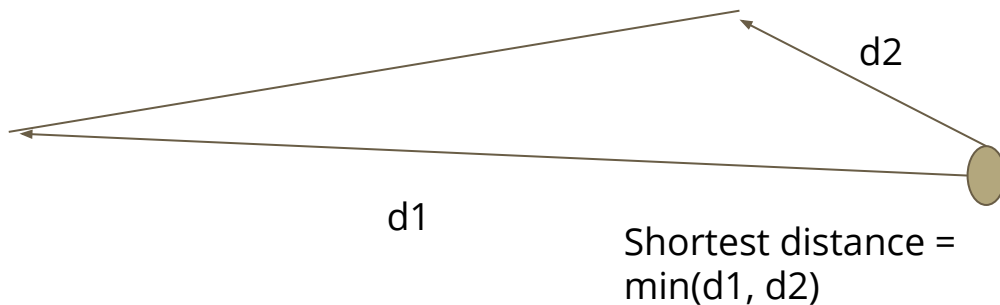
Method II Map matching- Each link is a sequence of control segment



1. The probability that the probe point belongs to each of the sublink is calculated. We call the line segment formed by joining two consecutive shape points of a link to be a sublink
2. Let P1 denote the probability that the probe point belongs to the sublink 1, P2 denote the probability that the probe point belongs to sublink 2 etc..
3. The probability that the probe point belongs to a **link** = **max(P1, P2, P3,...Pk)**

Map matching algorithm - Math details (Shortest distance of a point to a line segment)

1. We wanted to calculate shortest distance between the point and the line segment (of the sublink).
2. The shortest distance can take 2 forms
 - a. Perpendicular distance to the line segment if there exists a perpendicular line that contains P and intersects with the line segment
 - b. Minimum of the shortest distance to either end points of the line



Shortest distance =
Perpendicular distance
of the point from the
line

Map matching algorithm - Math details (Shortest distance of a point to a line segment)

1. First, find if a perpendicular line exists that intersects with the sublink line segment but also contains the probe point. This can be done by calculating the value of t . ($Vx1, Vy1$) and ($Vx2, Vy2$) stand for the ends points of the line segment and ($px1, py1$) stand for the point

$$t = -\frac{(vx1 - px)(vx2 - vx1) + (vy1 - py)(vy2 - vy1)}{(vx2 - vx1)^2 + (vy2 - vy1)^2}$$

2. If $0 \leq t \leq 1$, then find the perpendicular distance of a point to a line segment

$$d = \frac{|(vx2 - vx1)(vy1 - py) - (vy2 - vy1)(vx1 - px)|}{\sqrt{(vx2 - vx1)^2 + (vy2 - vy1)^2}}$$

3. If $t < 0$ or $t > 1$, find the distance of the point to two end points of the line segment. The shortest distance of the point to the line segment is the minimum of the two end point distances

Map matching algorithm - Probability calculation

1. All probabilities are assumed to be gaussian distributed
2. $P(p | k_i)$ = Probability that the probe point P belongs to link k_i

$$P(p|k_i) \propto P(b|k_i)P(\Delta\phi|k_i)$$
$$P(b|k_i) = \frac{1}{\sigma_B\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{b}{\sigma_B}\right)^2}$$
$$P(\Delta\phi|k_i) = \frac{1}{\sigma_\Phi\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\Delta\phi}{\sigma_\Phi}\right)^2}$$

3. The variance value for distance was taken to be 5 meters and the variance value for the heading was taken to be 1.042 (60 degree).
4. The heading for the road sublink was calculated from the angle of the line segment that joins two consecutive points
5. We tried to extend this to include the speed information as well by including a third probability term but we didn't know the information on how to convert speed category (1-8) to a speed limit of the lane. If this information is known, it becomes straightforward to extend this to include speed information here

Runtime Analysis

Method- I

1. Time taken to process one probe point - 74 ms
2. Time taken to process the whole data - 52h

Method - II

1. Time taken to process one probe point - 60 ms
2. Time taken to process the whole data - 42h

All timings are based on normal CPU performance (i8 quad core processor). The ability to do such high speed processing is mainly due to our database based architecture. Even in this pipeline, the algorithmic processing time is much smaller compared to db fetch time. A db fetch usually takes anywhere between 30 - 150 ms. The time per record was short in the beginning but slowed down heavily as the processing progressed. This must be due to the memory build up of the algorithm

Results

1. For calculating the accuracy, we downloaded the database as csv file and finally run our **“Evaluation script”** to check if the slope predictions given by the probe data lines with the gt slopes given in the data.
2. We used **the weighted slope calculation** suggested in the assignment to calculate a single slope value for a link (Slope of each sub-link segment is weighted by its length and average)
3. Finally we calculated the MSE (Mean square error) between the predicted slope and the actual slope across all the of the links.
4. We can make an end judgment as to what gave us the best results on the total data. Map matching method -2 along with the slope calculation algorithm gave us the best results.

Method-1 map matching Error	Method-2 map matching Error
0.43	0.34

5. One thing to be aware of is that it's hard to make a confident judgement on which is the better map matching algorithm purely through the slope calculation errors, since the the slope calculation error is not a direct metric for assessing map matching performance. It would more ideal to compare the performance of the two map matching algorithm with a data that had known map matching Ground Truth
6. Nevertheless, its expected that method-II of map matching will give better results for map matching and its indirectly reflected in the results through better slope accuracies for method-II.