Probe Data Analysis for Road Slope

Juhi Deshpande	A20430435
Sonali Deshpande	A20428066
Monishaa Arulalan	A20432700

Contents

1. Introduction
2. Tasks to be performed
3. Approach
4. Result
5. References

1. Introduction

- ❑ Probe Data : It is data which is generated by monitoring the position of moving objects over space and time. These moving objects can be dedicated vehicles used to collect information.
- ❑ Probe Data Format :
    - ❑ sampleID - is a unique identifier for the set of probe points that were collected from a particular phone.
    - ❑ dateTime - is the date and time that the probe point was collected.
    - ❑ sourceCode - is a unique identifier for the data supplier (13 = COMPANY).
    - ❑ Latitude - is the latitude in decimal degrees.
    - ❑ Longitude - is the longitude in decimal degrees.
    - ❑ Altitude - is the altitude in meters.
    - ❑ Speed - is the speed in KPH.
    - ❑ Heading - is the heading in degrees.

- Link Data : The link data is information of road segments (links). It consist of data for the links that probe points can be map-matched to.
- LinkData format :
  - linkPVID
  - refNodeID
  - nrefNodeID
  - Length
  - functionalClass
  - directionOfTravel
  - speedCategory
  - fromRefSpeedLimit
  - toRefSpeedLimit
  - fromRefNumLanes
  - toRefNumLanes
  - multiDigitized
  - Urban
  - timeZone
  - shapeInfo
  - curvatureInfo
  - slopeInfo

2. TASK TO BE PERFORMED

❑ In this assignment, we map match probe points to road links

❑ For each link found, we find the slope for each link

❑ Finally, we find the actual road slope and calculated road slope and evaluate them.

3. APPROACH

Step 1: Map-matching

❑ The co-ordinates contain the latitude and longitude of data in the Probe data and Link Data files.

❑ Those coordinates are then compared to every other pairs of values in the file to find the minimum one. The pair with the minimum distance is then further selected.

❑ The final pair is the one which is Map Matched Probe data and is further used as the corresponding link.

APPROACH CONTINUED.....

Step 2: Package the LinkData csv into a Package Object:

- ❑ Slope is calculated for two points by using matched points in previous step to calculate the slope of two consecutive points. The start and end points are passed to the class.
- ❑ The slope formula is (A1,A2):

Case 1: When latitude!=0

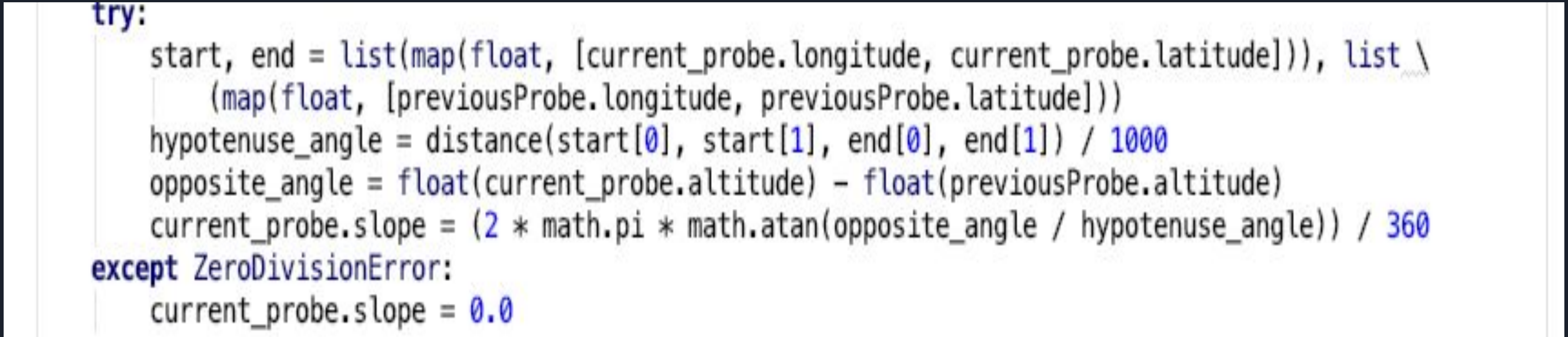
radian = math.atan(self.vector_longitude / self.vector_latitude)

Case 2: When longitude > 0

radian = math.pi / 2

Else : radian = math.pi * 3 / 2 and is calculated for all match points

CODE SNIPPET

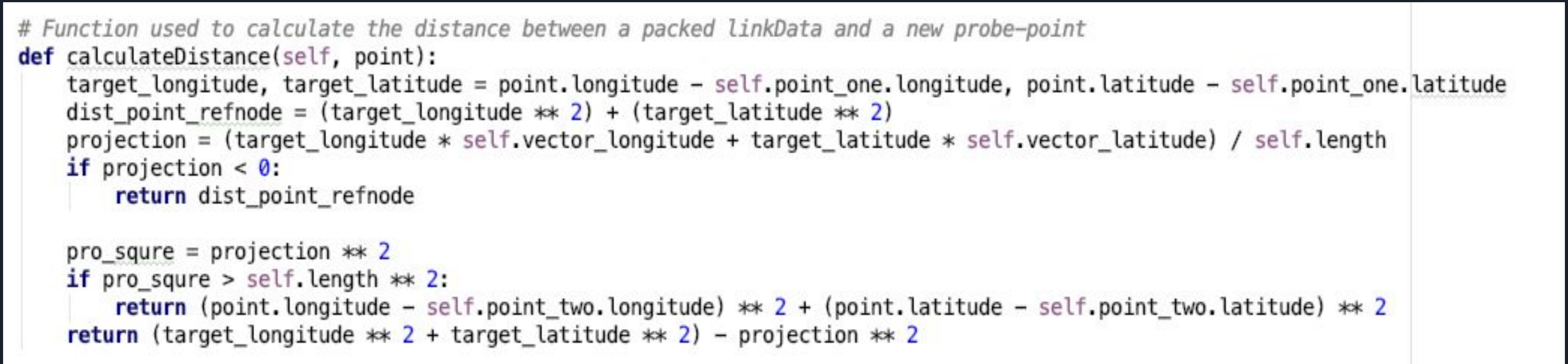


```
try:
    start, end = list(map(float, [current_probe.longitude, current_probe.latitude])), list \
        (map(float, [previousProbe.longitude, previousProbe.latitude]))
    hypotenuse_angle = distance(start[0], start[1], end[0], end[1]) / 1000
    opposite_angle = float(current_probe.altitude) - float(previousProbe.altitude)
    current_probe.slope = (2 * math.pi * math.atan(opposite_angle / hypotenuse_angle)) / 360
except ZeroDivisionError:
    current_probe.slope = 0.0
```

APPROACH CONTINUED.....

Step 3: Calculate the distance between a packed LinkData and a new probe-point:

Projection is calculated by using : projection = (target_longitude * self.vector_longitude + target_latitude * self.vector_latitude) / self.length



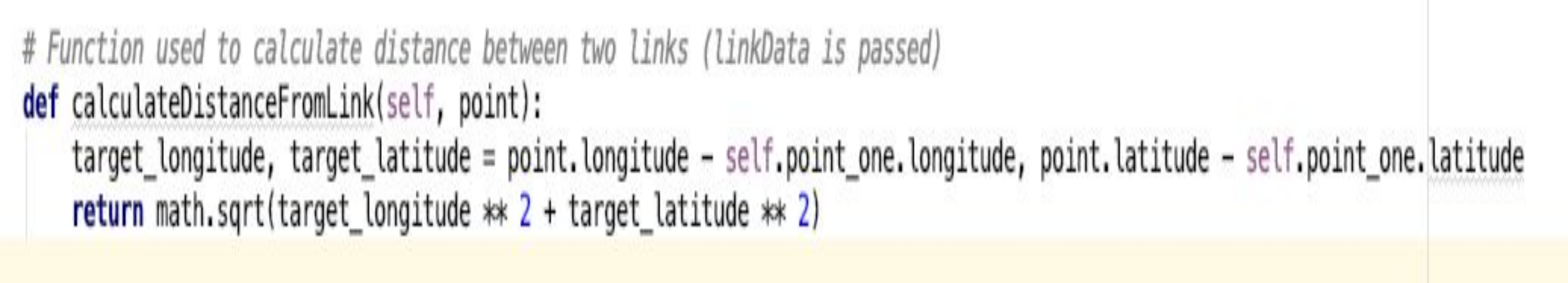
```
# Function used to calculate the distance between a packed linkData and a new probe-point
def calculateDistance(self, point):
    target_longitude, target_latitude = point.longitude - self.point_one.longitude, point.latitude - self.point_one.latitude
    dist_point_refnode = (target_longitude ** 2) + (target_latitude ** 2)
    projection = (target_longitude * self.vector_longitude + target_latitude * self.vector_latitude) / self.length
    if projection < 0:
        return dist_point_refnode

    pro_squre = projection ** 2
    if pro_squre > self.length ** 2:
        return (point.longitude - self.point_two.longitude) ** 2 + (point.latitude - self.point_two.latitude) ** 2
    return (target_longitude ** 2 + target_latitude ** 2) - projection ** 2
```

APPROACH CONTINUED.....

Step 4: Calculate distance between two links (linkData is passed):

Here we use the Euclidean distance formula to calculate the distance by using sqrt[(x2-x1)^2+(y2-y1)^2]



```python
# Function used to calculate distance between two links (linkData is passed)
def calculateDistanceFromLink(self, point):
    target_longitude, target_latitude = point.longitude - self.point_one.longitude, point.latitude - self.point_one.latitude
    return math.sqrt(target_longitude ** 2 + target_latitude ** 2)
```

APPROACH CONTINUED.....

Later on, the direction is calculated and the object is converted to string and the required format is returned with the scope appended.

Step 5: Calculate the slope and evaluate

- ❏ Used to read the linkData from the csv and create linkDataList/pointDataList
- ❏ Used to match the linkDataList with the probe data, find the shortest distance and create the Partition6467MatchedPoints.csv
- ❏ Used to distance between two data points (latitude and longitude) with respect to earth avg rad
- ❏ Used to find the slope of the road link
- ❏ Loop through matched point csv to find the slope and create slope csv
- ❏ Used to evaluate the derived road slope with the surveyed road slope in the link data file

4. RESULT

- The source code is present in the probe.py file and the ReadMe file contains the instructions as to how the program should be executed.
- Initially, the Partition6467MatchedPoints.csv is created and this .csv file is given as input for the slope calculation and evaluation.
- The result file named “evaluation” contains the final output.

5. REFERENCES

- Guest Lecture by Dr. Bo Xu
- “Online map-matching based on Hidden Markov model for real-time traffic sensing applications” by C.Y. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, P. Jaillet (2012 ITSC)
- “Map-Matching for Low-Sampling-Rate GPS Trajectories” Yu Zheng, MS Research
- “The Path Inference Filter: Model-Based Low Latency Map Matching of Probe Vehicle Data (Supplementary Materials)” Tim Hunter, Alex Bayern, University of California, Berkeley

THANK YOU