EE 495 Assignment 2: Probe Data Analysis for Road Slope

By: Christopher Tsai and Anuj Karnik

In [296]:
```
import numpy as np
import pandas as pd
import requests
import gmaps
import gmaps.datasets
gmaps.configure(api_key='INSERT_KEY')
from haversine import haversine, Unit
```

Visualizing ProbePoints in Pandas:

In [293]:
```
ProbePoints = pd.read_csv('Partition6467ProbePoints.csv', header=None, names=['sampleID', 'dateTime', 'sourceCode', 'latitude', 'longitude', 'altitude', 'speed', 'heading']
ProbePoints['dateTime'] = pd.to_datetime(ProbePoints['dateTime'], format='%m/%d/%Y %I:%M:%S %p')
ProbePoints.drop_duplicates(inplace=True)
ProbePoints
```

Out[293]:

	sampleID	dateTime	sourceCode	latitude	longitude	altitude	speed	heading
0	3496	2009-06-12 06:12:49	13	51.496868	9.386022	200	23	339
1	3496	2009-06-12 06:12:54	13	51.496682	9.386157	200	10	129
2	3496	2009-06-12 06:12:59	13	51.496705	9.386422	201	21	60
3	3496	2009-06-12 06:13:04	13	51.496749	9.386840	201	0	360
4	3496	2009-06-12 06:13:09	13	51.496864	9.387294	199	0	360
...	...	...	...	...	...	...	...	...
3375740	5840319	2010-03-01 05:55:35	13	52.217058	8.974134	130	100	271
3375741	5840319	2010-03-01 05:55:55	13	52.217080	8.966073	130	101	266
3375742	5840319	2010-03-01 05:56:27	13	52.214741	8.953245	136	104	248
3375743	5840319	2010-03-01 05:56:37	13	52.213855	8.949234	138	105	255
3375744	5840319	2010-03-01 05:57:17	13	52.211173	8.932944	138	106	248

3278652 rows × 8 columns

Visualizing LinkData in Pandas and converting "info" columns to list of lists:

In [264]:
```
LinkData = pd.read_csv('Partition6467LinkData.csv',  header=None, names=['linkPVID', 'refNodeID', 'nrefNodeID', 'length', 'functionalClass', 'directionOfTravel', 'speedCate
LinkData.fillna("0", inplace=True)

def infoToList(info):
    return [[float(j) for j in i.split('/')[:2]] for i in info.split('|')]  # [:2] in order to ignore latitude in shapeList

LinkData['shapeList'] = LinkData['shapeInfo'].apply(infoToList)
LinkData['slopeList'] = LinkData['slopeInfo'].apply(infoToList)
LinkData
```

Out[264]:

	linkPVID	refNodeID	nrefNodeID	length	functionalClass	directionOfTravel	speedCategory	fromRefSpeedLimit	toRefSpeedLimit	fromRefNumLanes	toRefNumLanes	Digitized	multiDigitized	timeZone
0	62007637	162844982	162809070	335.04	5	B	7	30	30	0	0	F	T	0.0
1	567329767	162844982	162981512	134.56	5	B	7	0	0	0	0	F	T	0.0
2	62007648	162877732	162844982	97.01	5	B	7	30	30	0	0	F	T	0.0
3	78670326	162877732	163152693	314.84	5	B	7	30	30	0	0	F	T	0.0
4	51881672	174713859	174587951	110.17	3	B	6	50	50	2	2	F	T	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200084	773675508	1470416664	174625689	64.34	4	B	7	30	30	1	1	F	T	0.0
200085	773675452	1470416431	174731801	82.29	4	B	7	30	30	1	1	F	T	0.0
200086	773675471	1470416516	174197627	104.72	4	B	6	50	50	1	1	F	T	0.0
200087	79691343	174771891	174833939	104.21	4	B	6	50	50	1	1	F	T	0.0
200088	79691344	174701000	174717482	124.73	4	B	6	50	50	1	1	F	T	0.0

200089 rows × 19 columns

Matching points to links by using minimum Great Circle distance from each point to links:

In [265]:

```
def linkMatching(lat, lon):
    probePoint = [lat, lon]
    tmp = LinkData[['linkPVID', 'shapeInfo', 'shapeList']]
    tmp['distFromLink'] = tmp['shapeInfo'].apply(lambda shapeInfo: min(haversine(probePoint, [float(j) for j in i.split('/')[:2]], unit=Unit.METERS) for i in shapeInfo.spli
    matchedLinkID = tmp.iloc[tmp['distFromLink'].idxmin()]['linkPVID']
    distFromLink = tmp['distFromLink'].min()
    indexOfLink = tmp[tmp['linkPVID'] == matchedLinkID].index.tolist()[0]
    refProbePoint = tmp['shapeList'][indexOfLink][0]
    distFromRef = haversine(probePoint, refProbePoint, unit=Unit.METERS)
    return matchedLinkID, distFromRef, distFromLink

columns = ['sampleID', 'dateTime', 'sourceCode', 'latitude', 'longitude', 'altitude', 'speed', 'heading', 'linkPVID', 'directionOfTravel', 'distFromRef', 'distFromLink']

MatchedPoints = pd.DataFrame(columns=columns)

N = 500    # N = ProbePoints.shape[0] = 3278652 takes way too long

for i in range(N):
    print("Matching probe point", i, "out of", N - 1)
    matchedLinkID = linkMatching(ProbePoints['latitude'][i], ProbePoints['longitude'][i])[0]
    distFromRef = linkMatching(ProbePoints['latitude'][i], ProbePoints['longitude'][i])[1]
    distFromLink = linkMatching(ProbePoints['latitude'][i], ProbePoints['longitude'][i])[2]
    indexOfLink = LinkData[LinkData['linkPVID'] == matchedLinkID].index.tolist()[0]
    matchedPoint = pd.DataFrame([[ProbePoints['sampleID'][i], ProbePoints['dateTime'][i], ProbePoints['sourceCode'][i], ProbePoints['latitude'][i], ProbePoints['longitude']
    MatchedPoints = MatchedPoints.append(matchedPoint)

MatchedPoints.to_csv('../Results/Partition6467MatchedPoints.csv', index=False)
MatchedPoints = MatchedPoints.reset_index(drop=True)
MatchedPoints
```

Visualizing MatchedPoints by grouping rows with respect to linkPVID:

In [266]:
```
MatchedPoints2 = MatchedPoints[['linkPVID', 'sampleID', 'dateTime', 'sourceCode', 'latitude', 'longitude', 'altitude', 'speed', 'heading', 'directionOfTravel', 'distFromRef
MatchedPoints3 = MatchedPoints2.groupby(['linkPVID']).apply(lambda MatchedPoints2: MatchedPoints2.sort_values(by=['linkPVID', 'sampleID', 'dateTime'])).drop(columns=['linkP
MatchedPoints3
```

Out[266]:

		sampleID	dateTime	sourceCode	latitude	longitude	altitude	speed	heading	directionOfTravel	distFromRef	distFromLink
linkPVID												
51865408	**147**	4553	2009-06-13 11:42:55	13	53.051923	8.807315	34	40	309	B	12.851820	12.851820
	148	4553	2009-06-13 11:43:00	13	53.052166	8.806392	33	53	291	B	78.754134	8.336686
	149	4553	2009-06-13 11:43:05	13	53.052412	8.805452	33	0	360	B	147.202331	60.461695
	356	4556	2009-06-13 08:29:57	13	53.052119	8.806147	52	56	119	B	91.546048	10.682700
	357	4556	2009-06-13 08:30:01	13	53.051856	8.806967	52	56	117	B	31.063158	31.063158
...	**...**	...	...	...	...	...	...	...	...	...	...	...
811768917	**91**	4552	2009-06-13 11:49:41	13	53.069274	8.798756	46	0	199	B	23.107349	1.235121
	92	4552	2009-06-13 11:49:46	13	53.069274	8.798756	46	0	177	B	23.081681	1.263576
	93	4552	2009-06-13 11:49:51	13	53.069274	8.798756	46	0	196	B	23.086928	1.263585
	94	4552	2009-06-13 11:49:56	13	53.069274	8.798756	46	0	190	B	23.088686	1.263776
	95	4552	2009-06-13 11:50:01	13	53.069279	8.798779	46	4	74	B	22.336300	2.583370

500 rows × 11 columns

Calculating slope of links by using distance and altitude difference of points that are matched to each link:

In [287]:
```
LinkedSlopes = pd.DataFrame(columns=['linkPVID', 'slopesList', 'averageSlope'])

for linkPVID, group in MatchedPoints2.groupby(['linkPVID']):   # groupby outputs a dictionary type
    slopesList = []
    for i in range(len(MatchedPoints3.loc[linkPVID]) - 1):
        probePoint1 = [MatchedPoints3.loc[linkPVID, :].reset_index(drop=True).loc[i, 'latitude'], MatchedPoints3.loc[linkPVID, :].reset_index(drop=True).loc[i, 'longitude']
        probePoint2 = [MatchedPoints3.loc[linkPVID, :].reset_index(drop=True).loc[i + 1, 'latitude'], MatchedPoints3.loc[linkPVID, :].reset_index(drop=True).loc[i + 1, 'lor
        distance = haversine(probePoint1, probePoint2, unit=Unit.METERS)
        changeInAltitude = MatchedPoints3.loc[linkPVID, :].reset_index(drop=True).loc[i + 1, 'altitude'] - MatchedPoints3.loc[linkPVID, :].reset_index(drop=True).loc[i, 'al
        slope = np.arctan(changeInAltitude/distance)
        slopesList.append(slope)
        averageSlope = sum(slopesList)/len(slopesList)
        linkedSlope = pd.DataFrame([[linkPVID, slopesList, averageSlope]], columns=['linkPVID','slopesList','averageSlope'])
        LinkedSlopes = LinkedSlopes.append(linkedSlope)

LinkedSlopes = LinkedSlopes.reset_index(drop=True)
LinkedSlopes['groundTruthAverageSlope'] = LinkData['slopeList'].apply(lambda slopeList: sum([slope[1] for slope in slopeList])/len(slopeList) if len(slopeList) > 1 else 0.0
LinkedSlopes['absAverageSlopeError'] = abs(LinkedSlopes['averageSlope'] - LinkedSlopes['groundTruthAverageSlope'])
LinkedSlopes.to_csv('../Results/Partition6467SlopeResults.csv', index=False)
LinkedSlopes
```

Out[287]:

	linkPVID	slopesList	averageSlope	groundTruthAverageSlope	absAverageSlopeError
0	51865408	[-0.014848655965818238, 0.0, 0.322861327837228...	-0.014849	0.000	0.014849
1	51865408	[-0.014848655965818238, 0.0, 0.322861327837228...	-0.007424	0.000	0.007424
2	51865408	[-0.014848655965818238, 0.0, 0.322861327837228...	0.102671	0.000	0.102671
3	51865408	[-0.014848655965818238, 0.0, 0.322861327837228...	0.077003	0.000	0.077003
4	51866677	[-0.04937019685324395, 1.108069157942699]	-0.049370	-0.014	0.035370
...	...	...	...	...	...
314	811768917	[0.0, -1.103392124292288, 0.0, 0.0, 0.0, 0.0, ...	-0.183899	0.000	0.183899
315	811768917	[0.0, -1.103392124292288, 0.0, 0.0, 0.0, 0.0, ...	-0.157627	0.000	0.157627
316	811768917	[0.0, -1.103392124292288, 0.0, 0.0, 0.0, 0.0, ...	-0.137924	0.000	0.137924
317	811768917	[0.0, -1.103392124292288, 0.0, 0.0, 0.0, 0.0, ...	-0.122599	0.000	0.122599
318	811768917	[0.0, -1.103392124292288, 0.0, 0.0, 0.0, 0.0, ...	-0.110339	0.000	0.110339

319 rows × 5 columns

Printing average of averages of slope error of each link:

In [289]:
```
print("Average absAverageSlopeError =", LinkedSlopes['absAverageSlopeError'].mean(), "degrees")
```

```
Average absAverageSlopeError = 0.24347067152685037 degrees
```

Visualizing probe point and link pairing in gmaps (using very small sample size of 5):

In [312]:
```
MatchedPoints4 = pd.DataFrame(columns=columns)

N2 = 5

for i in range(N2):
    matchedLinkID2 = linkMatching(ProbePoints['latitude'][i], ProbePoints['longitude'][i])[0]
    distFromRef2 = linkMatching(ProbePoints['latitude'][i], ProbePoints['longitude'][i])[1]
    distFromLink2 = linkMatching(ProbePoints['latitude'][i], ProbePoints['longitude'][i])[2]
    indexOfLink2 = LinkData[LinkData['linkPVID'] == matchedLinkID2].index.tolist()[0]
    matchedPoint2 = pd.DataFrame([[ProbePoints['sampleID'][i], ProbePoints['dateTime'][i], ProbePoints['sourceCode'][i], ProbePoints['latitude'][i], ProbePoints['longitude'
    MatchedPoints4 = MatchedPoints4.append(matchedPoint2)
```

```
<ipython-input-265-d691d6bd0038>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  tmp['distFromLink'] = tmp['shapeInfo'].apply(lambda shapeInfo: min(haversine(probePoint, [float(j) for j in i.split('/')[:2]], unit=Unit.METERS) for i in shapeInfo.spli
t('|')))
```

In [382]:

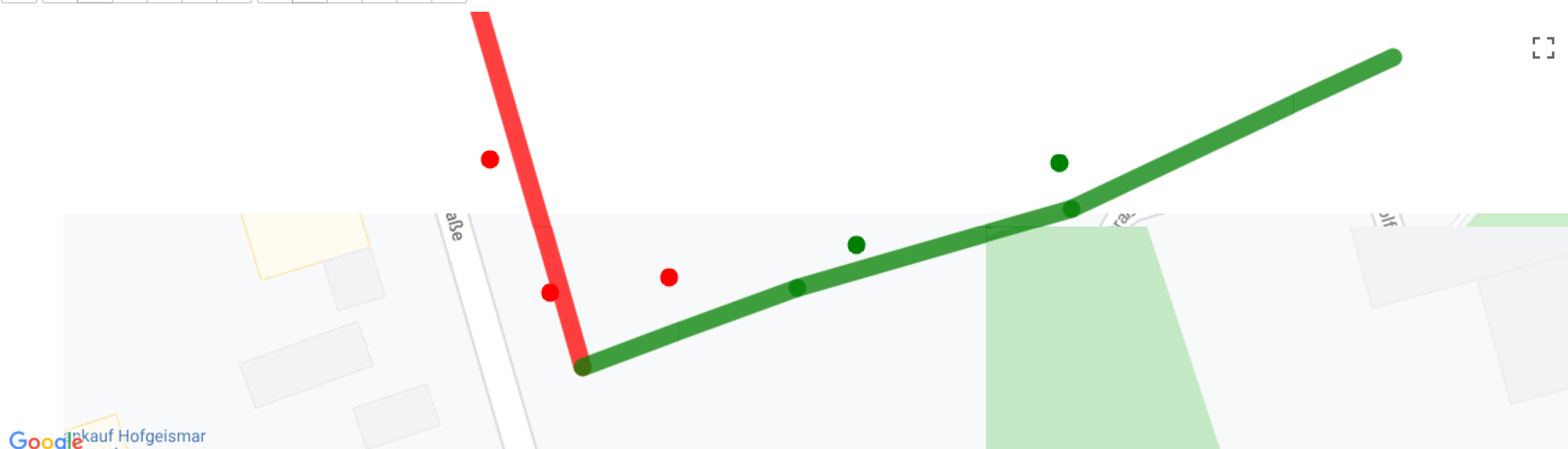
```
MatchedPoints4 = MatchedPoints4.reset_index(drop=True)
MatchedPoints5 = MatchedPoints4[['linkPVID', 'sampleID', 'dateTime', 'sourceCode', 'latitude', 'longitude', 'altitude', 'speed', 'heading', 'directionOfTravel', 'distFromRe
MatchedPoints6 = MatchedPoints5.groupby(['linkPVID']).apply(lambda MatchedPoints5: MatchedPoints5.sort_values(by=['linkPVID', 'sampleID', 'dateTime'])).drop(columns=['linkP

fig = gmaps.figure()

colors = ['red', 'green', 'blue']
color_idx = 0

for linkPVID, group in MatchedPoints5.groupby(['linkPVID']):
    pointsList = []
    linksList = []
    if (color_idx > len(colors)):
        color_idx = 0
    for i in range(len(LinkData[LinkData['linkPVID'] == linkPVID].reset_index(drop=True).loc[0, 'shapeList']) - 1):
        refNode = tuple(LinkData[LinkData['linkPVID'] == linkPVID].reset_index(drop=True).loc[0, 'shapeList'][i])
        nrefNode = tuple(LinkData[LinkData['linkPVID'] == linkPVID].reset_index(drop=True).loc[0, 'shapeList'][i + 1])
        link = gmaps.Line(start=refNode, end=nrefNode, stroke_weight=15.0, stroke_color=colors[color_idx], stroke_opacity=0.75)
        linksList.append(link)
    layer1 = gmaps.drawing_layer(features=linksList)
    fig.add_layer(layer1)
    for i in range(len(MatchedPoints6.loc[linkPVID])):
        pointsList.append([MatchedPoints6.loc[linkPVID, :].reset_index(drop=True).loc[i, 'latitude'], MatchedPoints6.loc[linkPVID, :].reset_index(drop=True).loc[i, 'longitu
    layer2 = gmaps.symbol_layer(pointsList, fill_color=colors[color_idx], stroke_color=colors[color_idx], scale=5)
    fig.add_layer(layer2)
    color_idx += 1

fig
```



(https://maps.google.com/maps?ll=51.496754,9.386695&z=19&t=m&hl=en-US&gl=US&mapclient=apiv3)

(
(