BDA FINAL PHASE SUBMISSION BY

VAISHNAVI BADAME AND

ANKIT JAIN

AIM:

To place place-markers at every latitude and longitude to detect a stop sign, detect left and right turns in the kml file generated, data cleaning and finally amongst the given files, find the file having the best cost function as per defined in previous checkpoint.

DATA CLEANING AND ANOMOLY DETCTION:

So, we have used RMC values mainly for our major calculations and tasks. So, these were some of the data cleaning aspects taken into account:

- The length of the RMC data values must be 13.
- The 10^{th} and 11^{th} attribute in the RMC data point is allowed to be empty.
- The other values cannot be empty as we require and are dependent on RMC for all data values.
- For the best route calculation, we need a file that has a minimum of 100 data points so as to find out a value for cost function that has a significant say. If a data file has a couple of points, the value turned out to be 0. So, a minimum threshold of points is set by us.
- The 2^{nd} attribute in RMC must be 'A' that indicates the signal is fine and has no issue. But if it is 'V', it indicates that there is navigation receiver warning. So, we avoid any data point that has 'V'.
- For the best route calculation, since we need the last line data for getting the time for calculating the overall trip time, the last line shouldn't be starting with 'lng', or it shouldn't be an empty line or any other data for gps such as '$GPVTG' or '$GPGSA'. It only considers the RMC or GGA as the last line to fetch the time. These are data points but not what we want in required format. So, anomaly.
- We have skipped the GGA or lng line in almost all calculations for simplicity as RMC itself was giving us all the required answers.

METHODOLOGY:

- **PLACE MARKERS:** we have only considered the RMC values as it has all the information required such as latitude, longitude and speed. So, once we have all these 3 calculated, we check at each data point for speed. If the speed is 0, that means the car has either stopped suddenly, or for a wait sign indicating it has halted. So, immediately then write a new kml file that will have the required tags for each data point with coordinates and speed value. Those points will have the place markers indication the car has stopped. The way we ran it is as follows:

initially, the file is converted to kml, then that file is dragged and dropped to Google earth pro for desktop that will show the path. Then the new kml that has new data points has to be separately dragged and dropped to Google earth pro to see the place-markers that have been generated. We have not used a loop for all the 173 files, but any file can be run on the code to show the respective stop sign markers. No specific data structure was used. Normal If statements and for loops have been used to detect the stop signs. There were no such issues at this task. It was pretty straight forward as professor in his write up mentioned the code on how to put a place marker using XML tags.

- **BEST ROUTE CALCULATION:** for this, we have cleaned data as mentioned above. We first calculate the overall trip time to know what the total time of travel is. Then for every data point, we assign the row a unique id for tracking purposes. For every speed value being 0, the corresponding timestamp of that record is added to a dictionary. The dictionary has the Id stored in a tuple as key and a list of timestamps in a list. For successive data points being 0, the last and 1st time stamps are subtracted to find wait time for that particular series of points. Then if another series has no speed being 0, continues till we find another successive points where speed is 0. Then all these differences for each series is added and makes the overall wait time. Each series of wait is given one unique id. Then this process is repeated for 173 files that are provided to us. For each file taken into consideration, we add the cost function value in a separate cost function list and its corresponding file name in a file name list. We then take the index of the minimum cost function and use that index to get the best file from the files list. At the end we have the lowest cost function and best file. This file indicates that this route has the least waiting time and as per the cost function is minimum. Some of the problems faced while coding for this were:
    - Getting the last line for fetching the time. This was handled by cleaning aspects that were added in the 'if-statement' as mentioned in the data cleaning part.
    - What will be the best data structure to store a unique id and a corresponding list that will be fast as we need to do the entire process for 173 files? So, we used a dictionary where the key is a tuple of ids and each unique id has a list that will have timestamp values.
    - How to assign id's as the wait time will not be continuous and will be irregular. This was handled by giving each series of speeds 0 as one single value using normal if statements and a unique id counter.
    - Iterating and indentation for 173 files.
- **TURN DETECTION:**

  We have only used RMC data and have not iterated over all 173 files. But any file can be run on this and we can get all markers where a right or left turn has been taken. So, the basic idea we have implemented is that when a person makes a turn, the angle does not change immediately and also the fact that we are getting data almost after every second. So, we take the 1st data point, and wait for the next 7 data points

(assuming the car takes some time to turn.). The old or first data points value is stored and then the last points value amongst the 7 values is passed to a 'turn detector function'. There we check the difference between the two passed data points angle. (Assumption: there is a turn if the angle is greater or equal to 30, ideally though a perfect left or right turn is 90 degrees.). Then if this value is less than 0 but the absolute value is greater than 30 degrees, then it is a left turn. This point will be written to another kml file with coordinates and speed value. Similarly, if the angle value is greater than 0 and the absolute value is greater or equal to 30, it is a right turn. (For a perfect left or right turn, the kml file will show a place marker, but for a smooth turn, it does not.). The way to run this code is to run the file as a kml first to get a path on Google earth pro. Then drag and drop another kml file of place-markers that indicate where the start point of left and right turns are. The left turn is indicated by a place-marker value of 3 and right by 4. No special data structures have been used, but simple if statements, for loops have been implemented to get this task done. Features used were angle of the start point and as per assumption, the angle of the 7th data point after the turn. Some issues were:

- How to decide a left or right turn. We used the angle value in RMC that tells us the magnetic deviation of the device or car with respect to the actual north of earth. Based on some online reading and logic, we were able to assign the start point of every turn.
- How to know when the turn starts or ends. Here we assumed that where ever the angle difference will be greater than 30 degrees, there is a possibility of a turn. Also the fact that it takes about 3-4 seconds to turn so, we could consider that point as our comparison to be sure it is an actual turn.

PROCEDURE TO RUN:

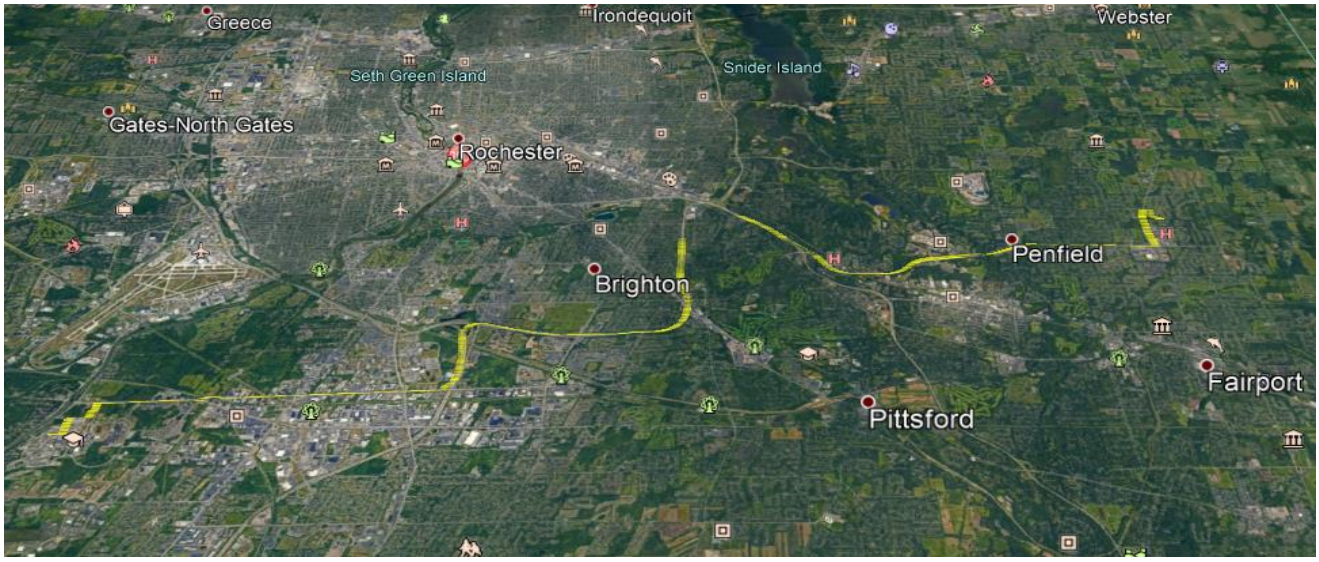
- Take a text gps file, run on checkPoints.py. Enter the particular filename along with txt as extension.
- It will generate 1 kml file, namely, GPS_Path.kml. This has the path.
- Then run turnDetection.py, that will generate 2 kml files, GPS_Hazards1.kml and GPS_Hazards2.kml. They have the turns and stop signs respectively.
- First drag and drop the GPS_Path.kml file and then drag the GPS_Hazards2.kml / GPS_Hazards1.kml as required file on that.
- For file with best cost function, run the bst_path.py file normally. It will return or display the best cost function along with the file name.

CONCLUSIONS AND LEARNINGS:

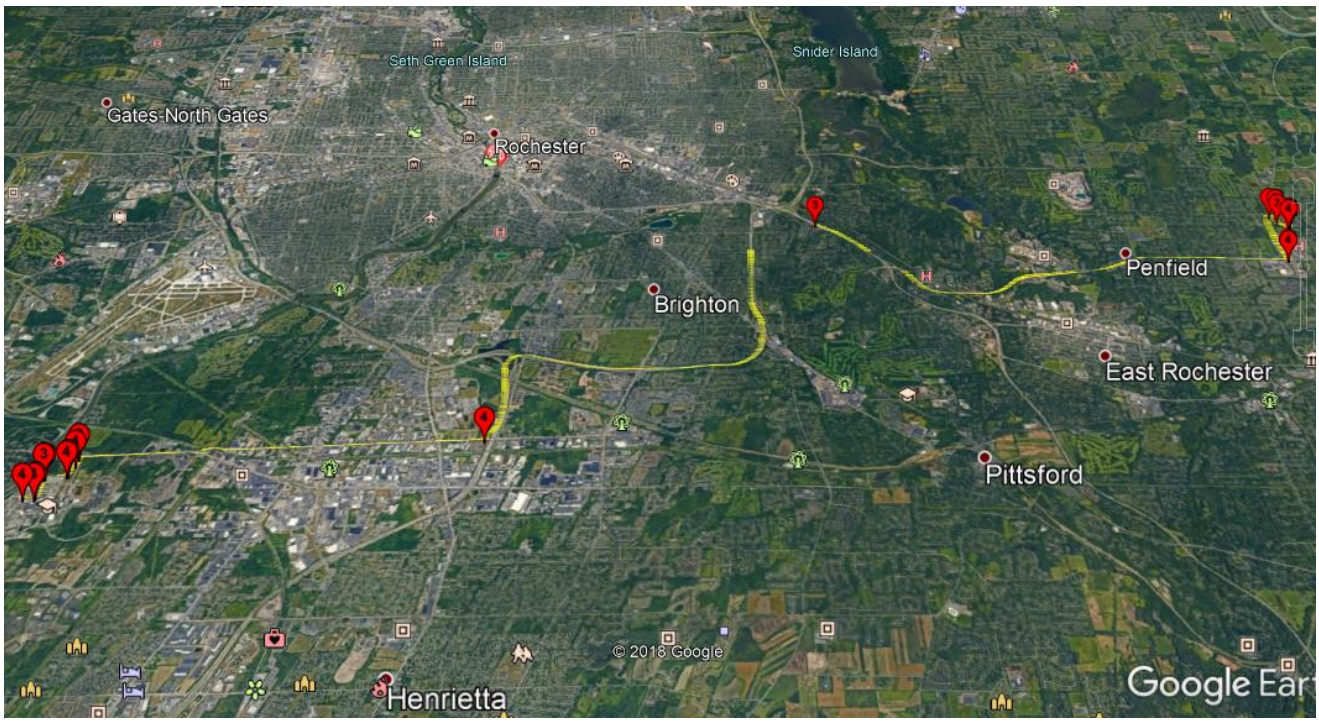
- Learn how to play around with GPS data.
- Learn how to use GPRMC and GPGGA data along with some differences and what attributes are in which one.
- Plot data on Google Earth Pro on desktop with given kml files.
- Analyse data and find best cost function.
- Use the 'datetime' library for time calculations.

- Read and manipulate many files in a single cod file all at once.
- Detect stop signs, left/right turns in gps data file based on certain parameters.
- Find the best file amongst the given files based on minimum cost function value.
- Write code that writes another code that has XML tags that finally can be rendered on Maps.
- Data cleaning for gps data for given format and specification.

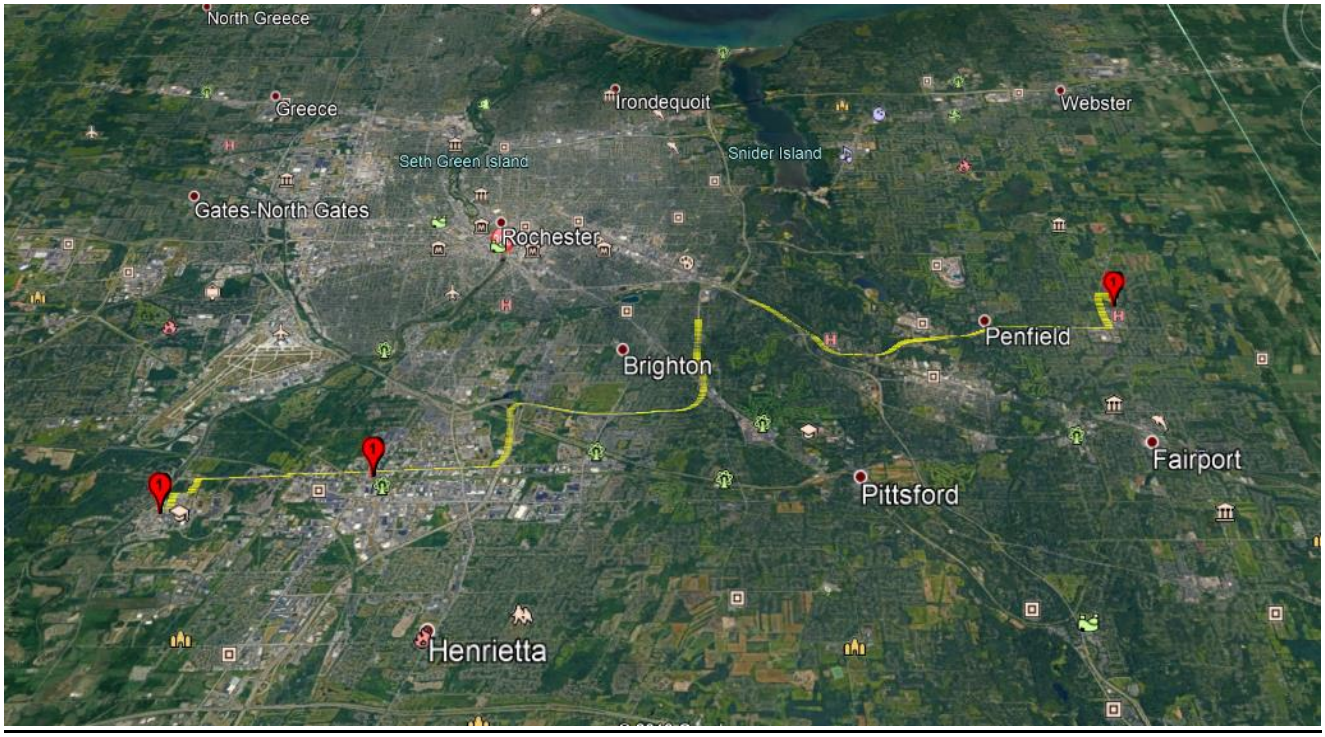
SCREEN SHOTS:



Path file.



Turn detection of left and right. Left is represented by a 3 sign and a right turn by a 4 sign. As we can see that for smooth curves or turns, as the speed could be slow, the turn wasn't detected.



Stop sign detection along with path. Indicated by 1.

BEST FILE AND MINIMUM COST FUNCTION:

The minimum cost function is: [0.019333333333333334]

C:/Users./ANKIT JAIN/Desktop/Masters Study Material/2nd SEM MATERIAL/Big data Analytics/GPS PROJECT/FILES/FILES_TO_WORK**2019_03_30__1718_23.txt**

NOTE: We had analysed some files at random and found out that the path is from professor's home and RIT. But some files have a different path or a totally different route all together. Request to take files that have source and destination as professor's home and RIT. We haven't filtered other files.