Description of work:

First of all, we are working with GPS data gathered by a Arduino UNO on different dates from Penfield to RIT and vice versa (It also consists of some trash files). The main motive is to find the best path that minimizes the cost function (number of left turns, stop signs, total trip time, etc) and creates a KML file for visualization.

⇒ We start with reading all the files from the data, then we perform certain data cleaning techniques on the data like, if a line consists of two segments one after the other then remove the other segment of data, deleting stationary coordinates from the data, removing the line of data where direction of the car is not changing for a while, removing data with Nan values, removing unwanted columns from the data frame, removing lines of data with incorrect end times, and many other.

⇒ Writing a cost function for minimizing the total trip time, number of left turns and stop signs. The file with the minimum cost value gets selected as the best path for travelling from RIT to Penfield and vice versa.

⇒ Writing a GPS to KML file which converts the (best path) GPS data file to kml for visualizing in map platforms (Google Earth) and keeps track of the left turns (Green Pins) and stop signs (Red Pins).

How you solved the problems

The rest of the report contains:

1. Conversion of GPS to KML and Data cleaning.
2. Cost Function.
3. Stoplight and turn detection.
4. Design of program.
5. Results.
6. Discussion.

To solve the problem, first, we had to clean up all the junk data and only consider valid data. Then we designed a cost function and calculated it. Finally, we converted GPS file with best route into KML.

Converting GPS to KML. What issues did you have converting the files? How did you do data cleaning and anomaly detection? What issues were in the files?

One of the issues we were facing was adding a different placemark for showing left turns (unlike the one provided in the question statement for stops signs). Our left turns were not visible initially, but it was due to a small mistake (I was using a non-compatible hex code). Data cleaning and Anomaly detection: We ran the code rigorously for all the text files provided by the professor and kept track of the cost value, time taken for the travel and the file name. We then found a threshold for the minimum time taken from RIT to Penfield and vice versa. This was important because the files consists of a lot of trash data (data where the start and destination are different from RIT or Penfield). While reading a file, we also keep a track of the latitude and longitude of start and end points to check if they are near RIT or Penfield (If they are not, we disregard those data points). Also, there are several times when some lines are repeated in the data one after the other (maybe due to parked vehicles), these duplicates are removed.

Final Cost function

Minimize:- 0.7*(Time taken) + 0.2*(Left turns) + 0.1*(Stop signs) (Objective) (Regularization)

Stop Detection: What classifier, and what features did you use to detect a stop or stoplight? What issues were there with it? How did you ignore the case when the vehicle was stopped to run an errand such as drop off a book at the library or go shopping?

Location and speed were the two features used to detect stops. One of the biggest issues while detecting stops was that many stops were being detected at the same location if the vehicle was stopped. This was solved by keeping track of the previous stop sign and calculating haversine distance from that point and if it was less than some threshold, we did not add the point to the list of stop signs.

Turn Detection: What classifier, and what features did you use to detect a left turn? How about a right turn? Where there any issues?

For the left turn detection, speed and the tracking angle were the two features used. We detected if the vehicle is slowing down or not and if it's slowing down, we check the current tracking angle with the previous angle and if the difference is greater than 25 we detected that its a left turn. Similarly, we could have done for a right turn by checking if the angle is less than -25.

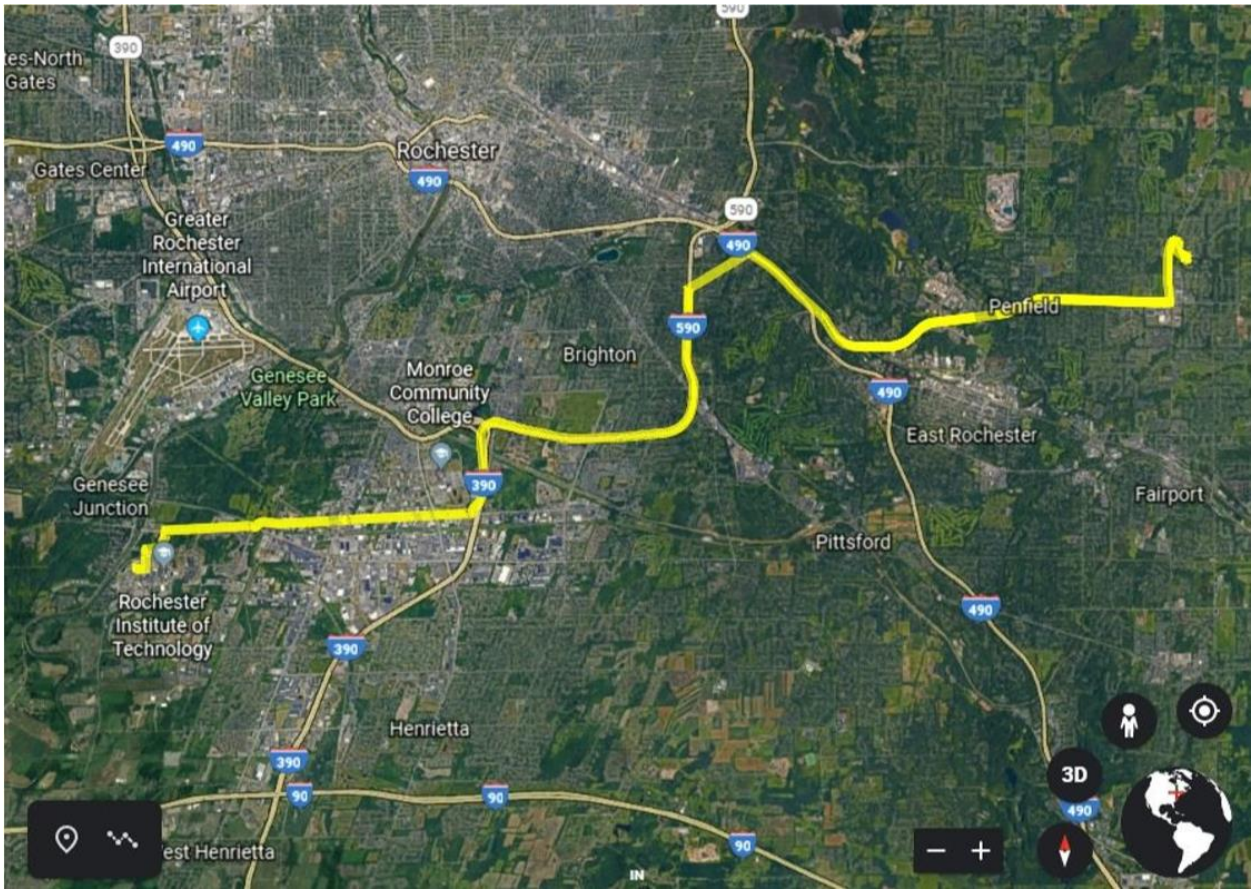
Describe how you designed and wrote your program. What design pattern did you use? What intermediate data structures did you use? What was your intermediate database? How did you handle assimilation across different tracks? What parameters did you use?

The program is basically divided into three parts, Data cleaning and preprocessing, finding the best route and converting GPS files to KML.

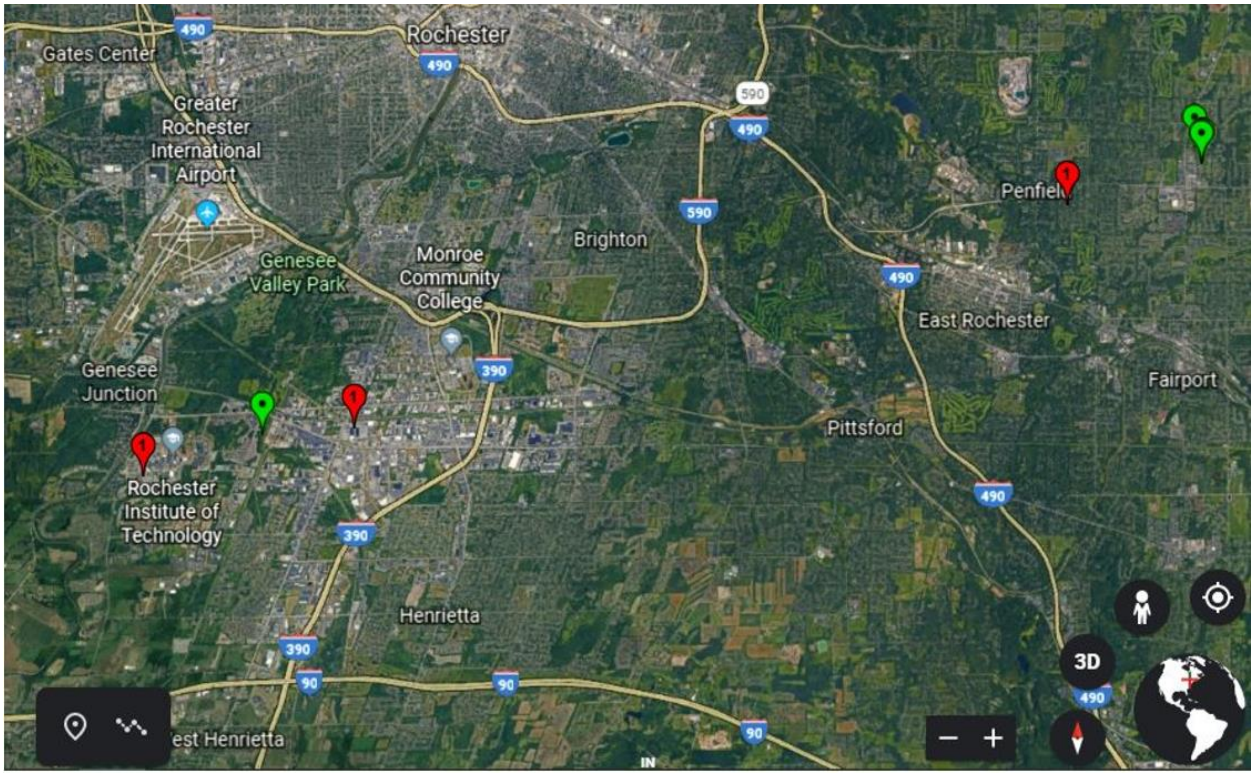
We first read all the data from .txt files and converted them into pandas data frames which are easier to work with. Every data frame was passed through a cleaning function that removed unwanted, erroneous and redundant data. For every data frame, the time taken was calculated along with a list of stop signs and left turns and using these three values, the best route was calculated.

Finally, a KML file was created using a self-made parser and stop signs and left turns were plotted in red and green respectively. The parameters used by this program are, Time, Latitude, Longitude, Speed, Tracking angle.

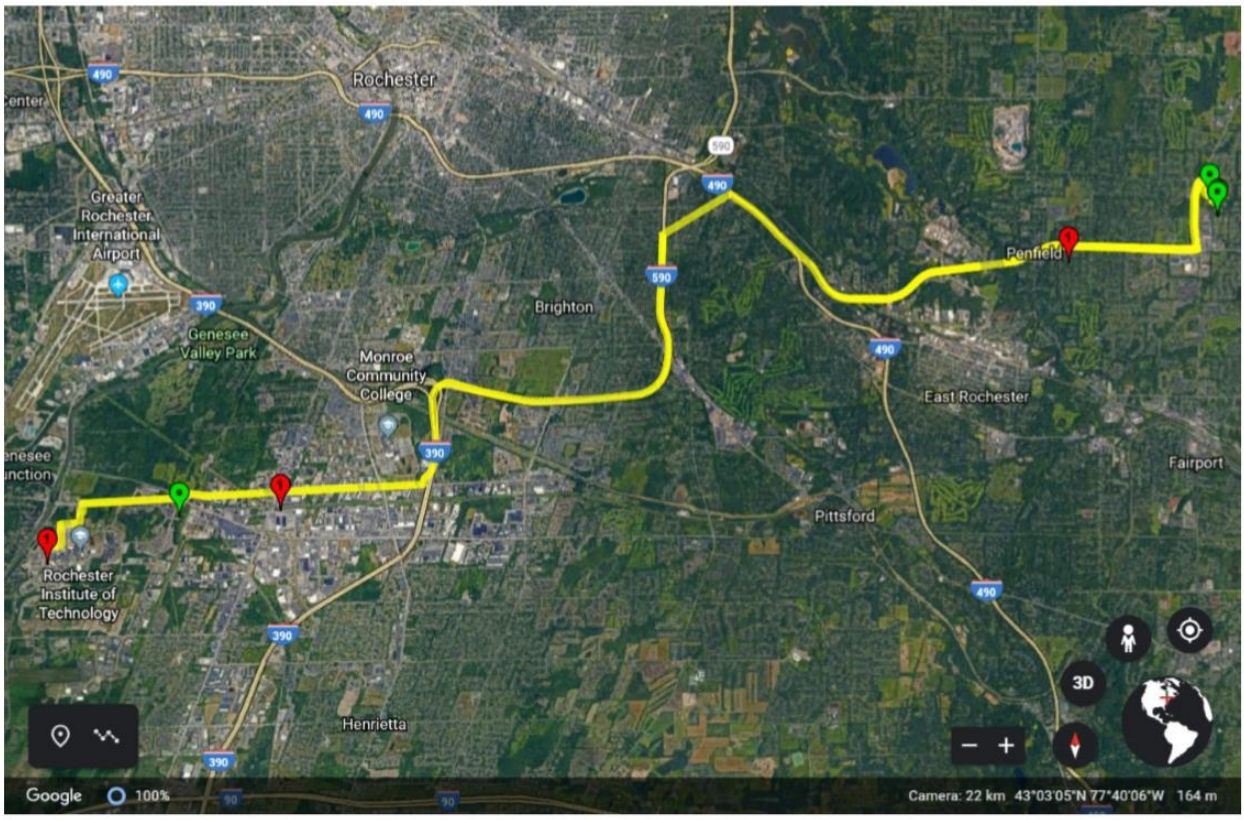
Results of a path file: Show an example screenshot of your path KML file.



Results of a hazard file: Show an example screenshot of your resulting turn and stop the KML file.



Screenshot of a *.kml file you created



What problems did you find with the approach? Did you have to do any noise removal or signal processing?

Data cleaning was one of the biggest problem with our solution. There was a lot of junk in the txt files as well as some files itself were a junk. Many times the resulting KML file was not from Penfield to RIT but some short trip in Penfield. Removing this was the biggest task. We handled this by only using the files whose time taken was greater than some threshold and finally, we found our solution.