GPS PROJECT

Abdul Hakim Shanavas
Maha Krishnan Krishnan

b. One Paragraph abstract which describes what you did.

We started with understanding the GPS data provided to us. The slides helped us understand each type of gps data and what are the fields that goes with it. This was the first step. Once we were comfortable with the input of our gps data, we went on to read each line and create a kml file showing the path travelled. We did not do any preprocessing of the data when we first created our first kml file. Hence, there were a lot of redundant data and too many coordinates. Next, it was the preprocessing step. Conversation with professor on the anomalies of gps data helped us do the clean our data. We decided to use $GPRMC gps data as it had all the required attributes for designing our classifier. We skipped lines of gps data which had two gps data in one line (burping of gps data), we saved only one gps coordinates for every second. From, analyzing the data we parsed, we found that there were many coordinates recorded for one second, which was not necessary. Deduplicated our latitude and longitude data. Then, designed a classifier to find the stop signs in the travelled path. Furthermore, designed classifier to detect the left and right turns in the path. We tried several classifiers and finalized on the classifier which worked best for the provided input data from visualization. Finally, we used agglomerative clustering to get rid of bunch of gps data classified as turns or stops and to make the final output look more elegant. After agglomerating, we deleted stops which was stopped due to a turn as mentioned in the description.

c. Overview section, which describes what is in the rest of the report. Give an overview of how you solved the problems.

Rest of the report talks in more depth about the classfiers we built, specifics about how did we define a turn and the challenges we faced along the project, limitations of our built model, team effort and spirit. Finally summarizing our entire experience of working on this gps project and what we learned from this project assimilating the knowledge gained from classroom setting and applying on this project.

We solved most of the problems, by understanding and analyzing the data. By analyzing I mean, by visualizing the data. For instance, we checked the angle between two coordinates of a right and left turn from the path kml file which gave us an idea of which angle to use for detecting turns in the classifier. Furthermore, visualizing the data helped us understand to decide on the number of clusters required to make the final kml to look more elegant by choosing one single point from the cluster of classified points.

d. Team composition: What role(s) did each team member have.
How did you balance the load across members?

Since we had to test a lot of data and to try different classifiers and fine tuning the classifier, both of us tried different classifier and cross referenced the kml file we got from the resulting classifier. This greatly helped us zero in on the classifier which worked best for our data. Furthermore, it was also challenging, when we had to do the batch processing. Hence, as we progressed, we saved our intermediate output to a file which was then shared among us to proceed to the next step (For instance, passing the stops and turns output to agglomerate from the data generated so far). So, both of our minds were put into task to make the code run faster and work well with the data. So, both of us had worked in every aspect of the project by trying out different ways of the same step.

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

To start off with, we had too much redundant data and so many gps coordinates for one second which was too huge. Especially, this affected when did batch processing. Furthermore, we cleaned the data by catching the lines in .txt file which had burped two gps data in one single line. We ignored those points by counting the occurrences of $GPRMC substring in each line. We also recorded only one coordinate for a second. After doing so, we dropped redundant data, that is deduplicated our dataframe based on the 'Latitude' and 'Longitude' column. Furthermore, the gps data such as 'Latitude' and 'Longitude' was not in the format which we required to generate kml file and the time was given in UTC plus the speed was given in nautical miles. We did the necessary conversion and saved the necessary fields into our dataframe with appropriate data type and format. This cleaning of data, helped us greatly increase the speed of our program and the efficiency of our classifier.

For example: The following file: 2019_03_04__Parked_at_RIT.txt had more than 80k lines of gps coordinates which was mostly redundant. We reduced that many coordinates or lines of data to 218 rows of data. Which is huge, because processing that many coordinates for turn and stop detection would have taken a lot of time which also degrades the classifier.

f. Stop Detection:
What classifier, and what features did you use to detect a stop or stop light?

We used attributes such as speed, latitude, longitude and the distance calculated from the latitude and longitude to design our classifier. The distance metric used was haversine distance. As said in the description, we filtered coordinates which had speed less than 10 miles per hour and classified a coordinate as a stop sign which had a distance of threshold = 5 meters from the previous stop to avoid pinning the same stop repeatedly.

What issues were there with it?

When a car was parked for a long time, there were a bunch of gps data which was classified at the same location despite cleaning the data. Hard to predict and ignore the stop where the vehicle stopped to run an errand. Also, initially a point was classified as stop even when it was stopped to take a turn. This was removed at the last step after agglomeration by removing stops which was in the range of a threshold = 100 meters of a turn sign.

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?

We tried to ignore those points by considering the time it was stopped, however, it was still difficult to detect those entirely as each point was processed and classified. Therefore, our classifier does not do well for differentiating a stop light and stop for running an errand. This is a limitation of our classifier, which will be our future work.

g. Turn Detection:
What classifier, and what features did you use to detect a left turn? How about a right turn?

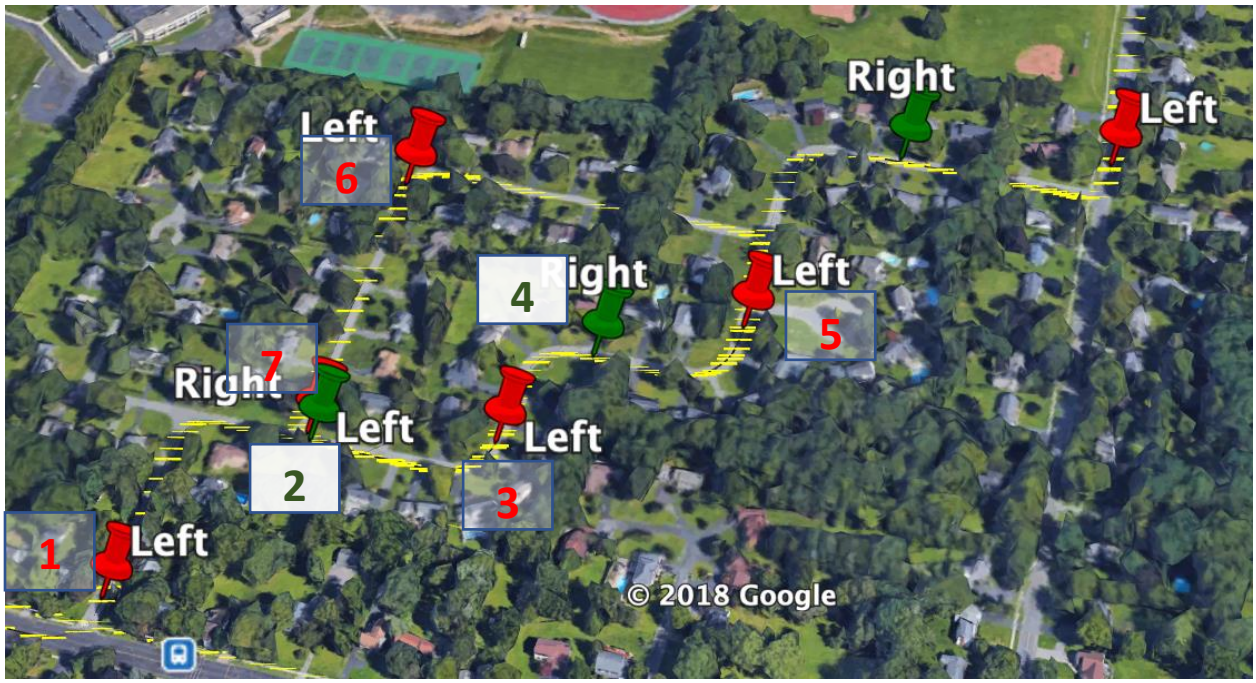
We tried several classifiers to detect turns. This was challenging, we tried strategies like finding angle between two coordinates and by using the angle in the gps data and doing subtraction. All this was done using a sliding window, and we tried different window size to check which one works for the data. We also tried by detecting points which had a speed less than a threshold and also like the difference in time of two coordinates as the car slows down when turning.

Finally, what worked for us was the angle in the gps data and by taking the difference in the angle and we manually found the range which falls in the turn by checking the angles of two coordinates in a turn in the kml file uploaded in google earth.

For a left turn, the difference of the angle between the range of 55 to 100 was classified as the left turn. For a right turn, the difference of the angle between the range of -50 to -90 and 260 to 300 was classified as right turn. By checking the kml files, we figured that for a right turn it falls into two ranges unlike a left turn. We fixated on the sliding window of 10 coordinates as a great deal of coordinates was deleted in data cleaning.

Where there any issues?

Yes, deciding on the range of the angle of difference between the coordinates and fixating on the sliding window range was challenging. It took us a considerable amount of analyzing the data to zero in on the range. Especially for a right turn, there were a lot of misses. When the range is wide, there were lot of false alarms and when the range was kept short there were a lot of misses. To balance out the false alarms and misses which works for the batch of .txt files was a challenging task.



The above two screenshots were taken from converting the 2019_03_03__RIT_to_HOME.txt file to kml. As we can notice that path travelled starting with a

1. Left (Red - Numbered)
2. Right (Green - Numbered)
3. Left
4. Right
5. Left
6. Left

7. Left

Points 2 to 6 completes a loop.

h. Assimilating across tracks:

Describe how you designed and wrote your program. What design pattern did you use?

We used batch processing programming pattern, ie, read in all data and process all data.

What intermediate data structures did you use? Did you build intermediate databases?

We saved all our stop sign, left and right turn coordinates in a separate text file after we did preprocessing and classification. This file was later read into as pandas dataframe and then converted to numpy array for doing agglomeration and removing stop signs which collided with a turn.

Did you hold all the information in memory at once? Did you parse all of the files at once?

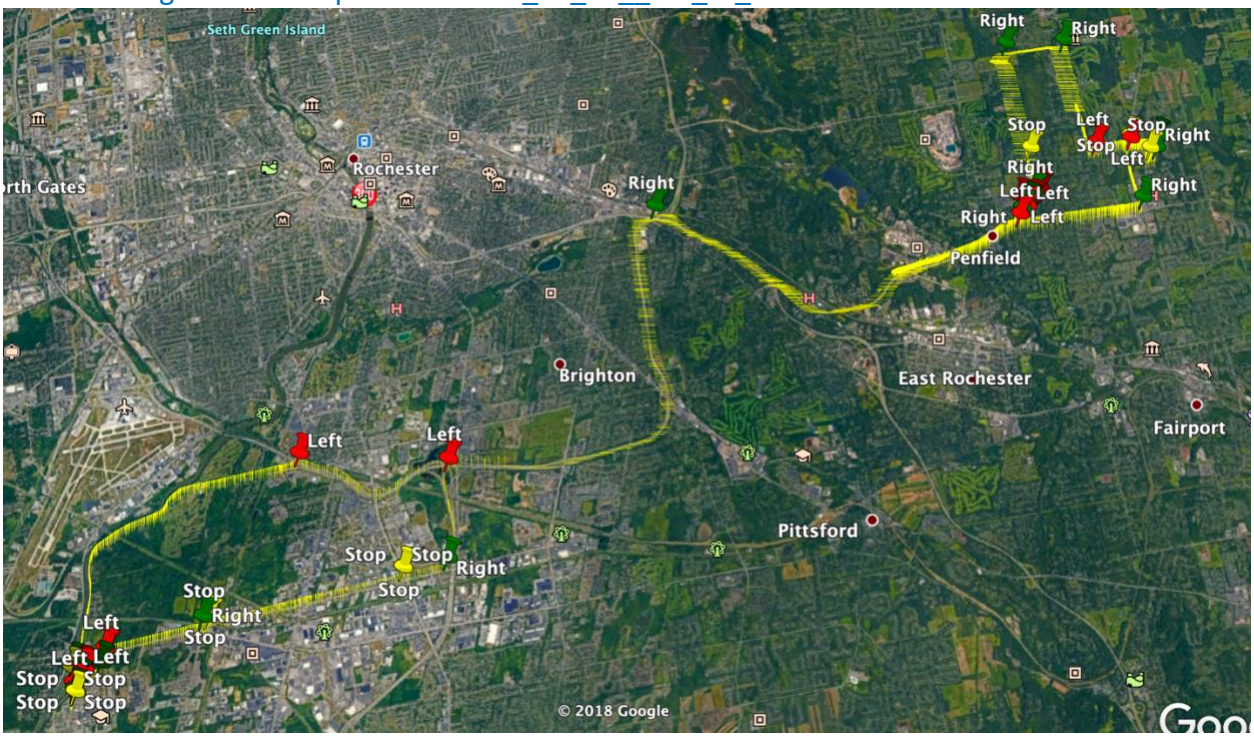
As every file was read, it was preprocessed, detected stop and turns and created a kml path file with all the signs and then all those coordinates was saved in a separate file to later use those coordinates for agglomeration and removing the stops colliding with turns. Therefore, yes it was process all at once and was hold retained in memory.

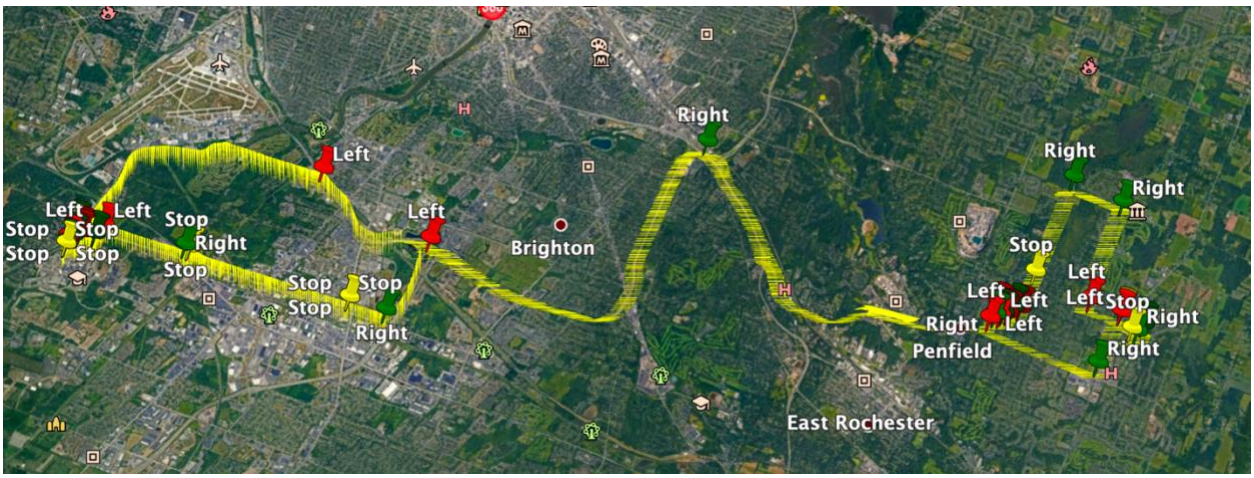
How did you handle assimilation across different tracks? What parameters did you use?

We stored all the stop and turn coordinates into one txt file each for stop, left and right. Read all those coordinates into pandas dataframe and converted to numpy array and performed agglomeration. We used haversine distance metric and single linkage for agglomeration with 80 clusters. Tried different clusters which worked well for our data.

i. Results of a path file: Show an example screen shot of your path KML file.

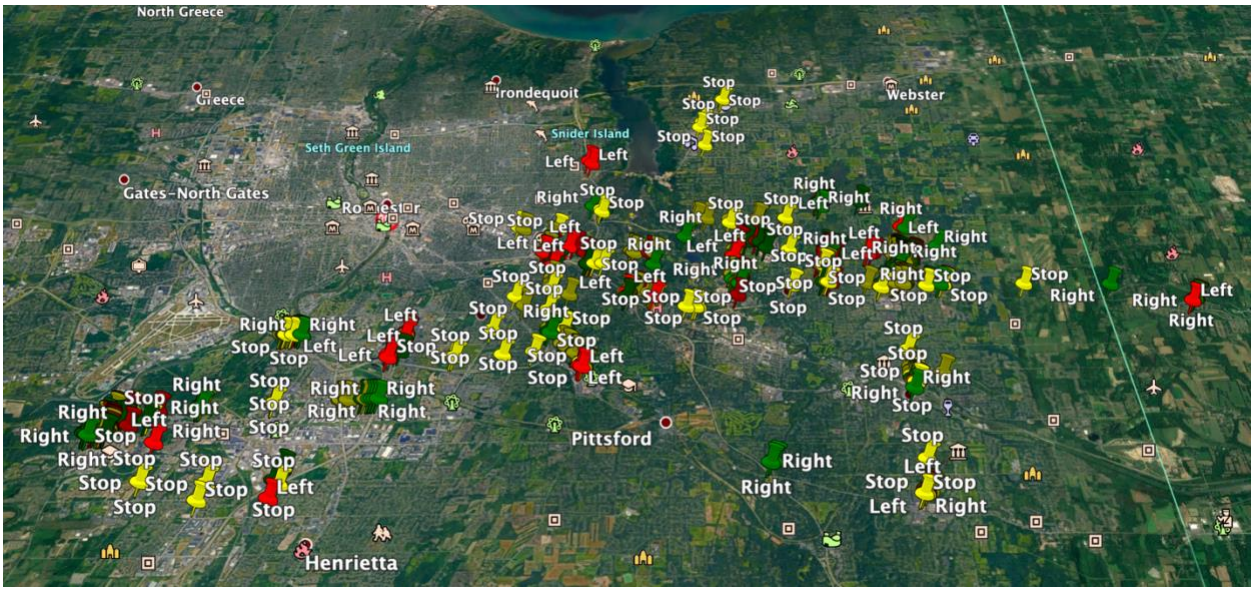
File used to generate the path file: 2019_03_03__RIT_to_HOME.txt



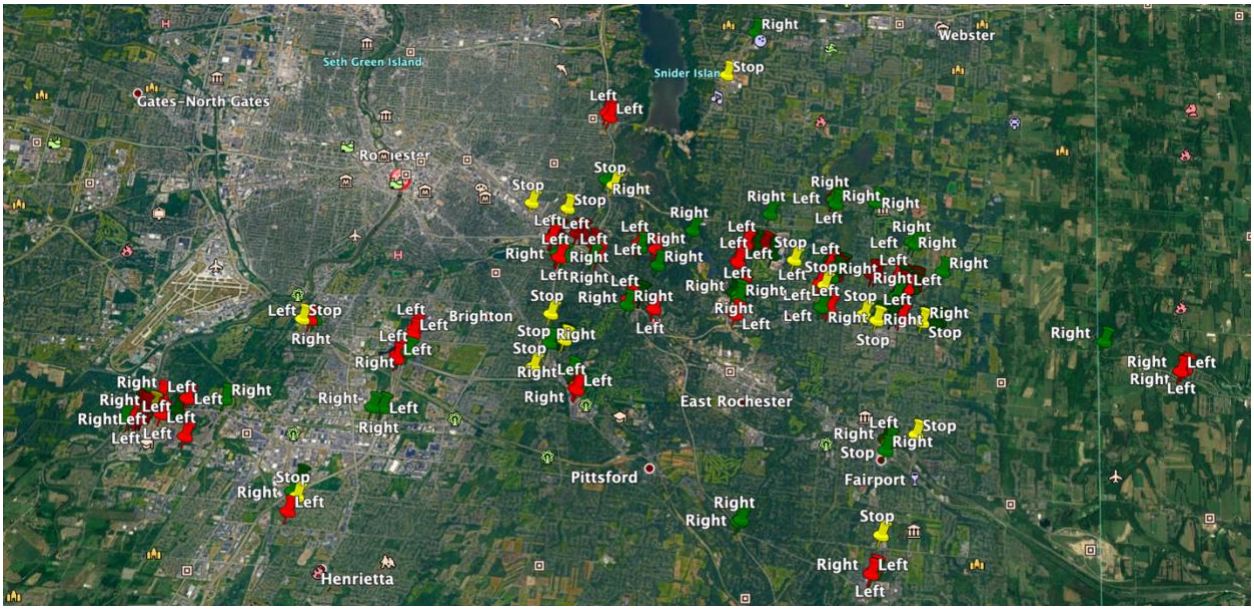


j. Results of a hazard file: Show an example screen shot of your resulting turn and stop KML file.

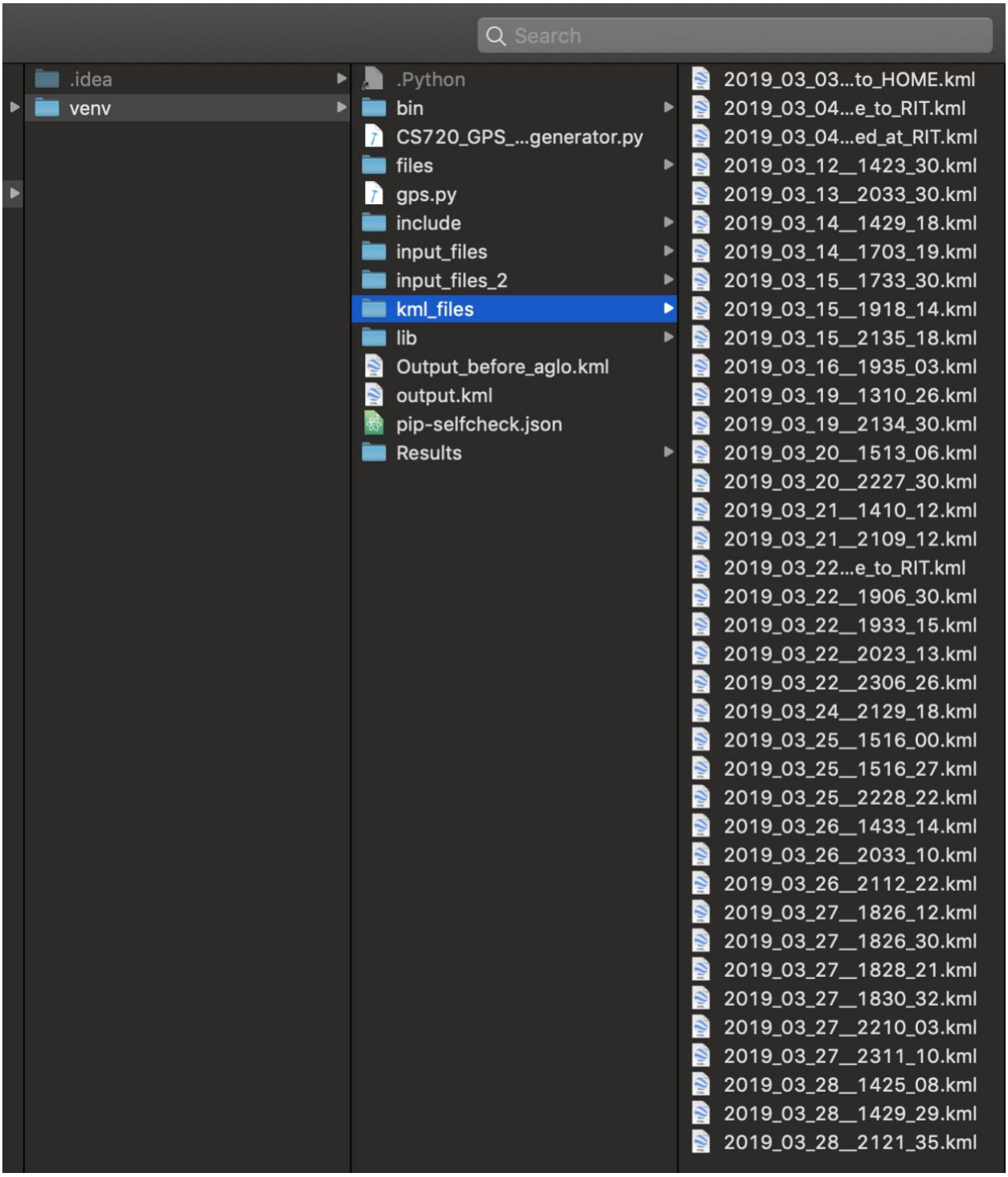
Before Agglomeration



After Agglomeration and removing stops which collided with turn.



k.Screen shot of a *.kml file you created, showing



l. Discussion section: What problems did you find with the approach? Did you have to do any noise removal or signal processing?

First, we did a mistake by not saving the dataframe after preprocessing the data, which was a big mistake as the program took a little while to run from start and create all files. Yes we had to remove noise removal such as too many coordinates when parked for a long time, stops classified when it was a turn.

m. A summary conclusion of what you learned overall, and how this might be useful for some commercial application.

We learned a great deal from this project. This project was a taste of a commercial application. We believe that this project will definitely stand out in our resume and we are also quite confident in what we did in this project. Even though this project was not perfect, I believe we learned and it made us think about some real time issues that we would face in the future.

We learnt how data preprocessing (Cleaning) is a very important step in terms of building a stable classifier with decent quality and also in terms of computation and time needed. Next thing is data visualization helped us a great deal in designing the classifier and choosing threshold and selecting features. All these steps were taught in the class day 1 in data mining steps. However, implementing those steps was fun and was great source of learning.