IoT Processing of GPS Data

CSCI 420

Jennifer Liu

Joseph Golden

Niccolo Dechicchio

Abstract

The main idea of the project is to visualize the paths a car has traveled using Google Earth and determine where the car stopped and took turns at. Given data gathered from an Arduino GPS attached to a car, we determine the path by converting GPS data files to a KML file in order to visualize the data on Google Earth. The area of travel was around the Rochester area to make things easier to visualize.

Overview

We were given 3 different tasks:

Task 1 - to estimate when a car stopped either due to a stop sign, due to traffic light, or due to parking.

Task 2 - when a vehicle turns left or right. The assumption is that slight road curvature are not included.

Task 3 - given multiple GPS paths, we had to agglomerate the paths into one.

In this paper, we will discuss about how we used clustering techniques to agglomerate paths and locate potential stopping locations. Furthermore, we will also discuss how we used decision-tree-like classifier to determine when a vehicle is turning left and right. The discussion will include the types of decision our classifier asked to determine these stops, and will also include the types of parameters for clustering.

GPS Text Data to KML

Figure 1.0 A single KML Path of all GPS data raw files provided by Dr. Kinsman.

Why are paths not connected?

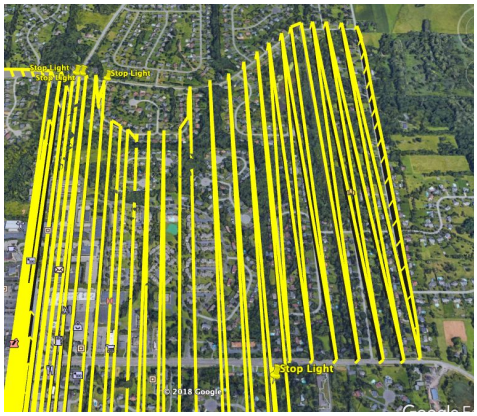

Figure 2.0 Line String failures. When we clustered, it causes points to be jumbled around. Therefore, connecting line string would not work.

We noticed that line string tries to connect points that are in sequential order. Since we clustered the points, the data is no longer in the order that it was given. Hence, the lines were zigzagging across the map. To address this issue, we created a line segment for every point. That is why the resulting KML paths are not connected, but more individual points.

Assimilating Paths

Agglomerating Paths - Removing Path Cluttering

When combining multiple GPS paths from different vehicle tracks, we noticed that many of these vehicles travelled on the same path. As a result, our KML files would generate points in almost identical locations, but was 2-3 meters off, hence creating a clutter of paths. **Figure 3.0** illustrates this problem.

Figure 3.0 Left Photo - Plotting raw GPS points before clustering. Right Photo - Plotting GPS points after clustering.

Because there were so many duplicated data points, when rendering the KML file, it was very laggy and took a really long time to render. To address this problem, we tried using agglomeration and K-means to cluster nearby points. We were able to throw away about ~60% of the data points, and still preserve relevant informations. We found that agglomeration didn't work as well as K-Means because agglomeration was not able to separate the GPS points nearly as well as K-Means can. After all, agglomeration is designed to allow for soft-mixture model. Whereas, K-Means was designed for hard-mixture model, meaning a point can only be in one cluster. We used Manhattan distance as the distance metric for K-Means. We figured that since we are not calculating distances across countries, but rather we were calculating distances between streets, manhattan distance seemed more appropriate. Through our testing, we tried an array of different numbers of clusters. After multiple trial and errors, we discovered that when combining all of the paths given by Professor Kinsman, which has ~60,000 points, 5000 clusters

were enough to render the path and not lose too much data. In fact, when boosting the clusters to 6000 and beyond, it had little effect on the path. The path started fading away as you zoom really far out, but when observing it up close visually, it still looked the same. Using the Ocam Razor's theorem, we went for the 5000 clusters because it was the simpler model.

Stop Detection

Agglomerating Stops - Fixing Dilution of Precision

Upon inspecting the paths, we noticed that when a car is parked, it tends to generate GPS data that is near the vicinity of where the car is parked, but the direction of the location can vary significantly. Hence, it generates path shown in **Figure 4.0**. To remove this, we checked when a car's speed is <= 0.5 mph, and if the distances of the current location and previous location is smaller than or equal 10 meters, then we remove that GPS point. (Refer to **Figure 4.0** for final results)

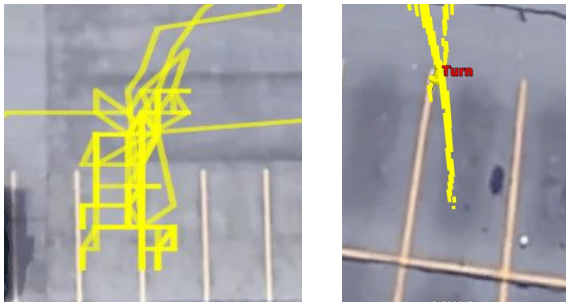

Figure 4.0 Left Photo, before stop agglomeration was applied. Right Photo, after stop agglomeration was applied

Upon analysing the paths, we noticed that when a car stop for a short period of time, it tends generate data points that are closer to each, and the data points are much denser in these regions. Whereas, when a vehicle is traveling in a straight line, the points are separated in a consistent manner and much more sparse. (Refer to **Figure 5.0** for evidence).

Figure 5.0 Left Photo - Data point pattern when a vehicle is near a stop light. Right photo - when a vehicle is traveling on a straight road.

Using this observation, we decided to use DBScan when classifying for stop signs. If there are 15 (min point) or more data points within a 0.0001 (epsilon) radius, we consider it a stop sign.

<u>Stop Detection Results</u>

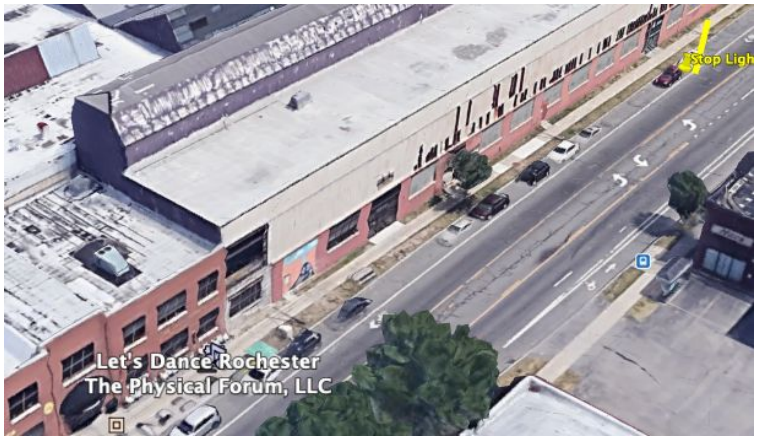

Figure 6.0 Some examples of our classifier getting a stop sign correct. It was able to detect that Dr. Kinsman parallel parked near **Let's Dance Rochester The Physical Forum, LLC** :)

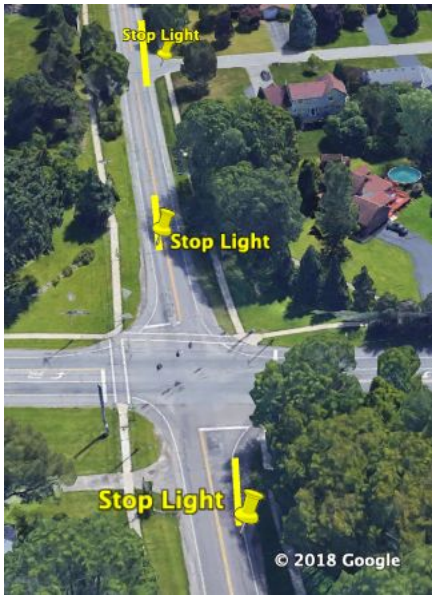

Figure 7.0 The classifier was able to detect stop signs that are super close to each other. It was also able to detect stop signs that are going in opposite directions due to different paths from different GPS raw files.

Figure 8.0 Stop detection occasional hiccups.

The image above showed that there are two stop lights super close to each other. However, the ground truth is that there is only 1. We attribute this to the fact that the epsilon radius is quite small (0.0001). Perhaps, if we increased this distance, it will remove this. However, it takes a really long time to cluster all the paths and it burns the hardware on our computer, which is not effective.

Stop Detection Shortcomings

We noticed that, when there were more paths, the classifier was doing better. With only single file paths, the classifier misses a lot of stops.

Team Compositions

Lead Developer: Jennifer Liu - Constructed the GPS to KML processes, developed methods for stop detection, noise removal, and classification of data points.Aided in documentation of the code as well.

Developer: Joseph Golden - aided in stop and turn detection classifiers as well as documentation of the code.

SQA, Documentation, Writeup: Niccolo Dechicchio - Aided in Documentation of code, writeup,and some SQA within the code.

Turn Detection

Our classifier for turn detection was a simple decision stub, differing only from a regular decision stub in that it uses a range of values instead of a single threshold. The attribute it checks is the absolute value of the change in angle between the current point and the next. If it is above 20 and below 170 it considers it a turn (negative is left, positive is right). These numbers seem not particularly intuitive considering we'd probably assume a "turn" to be a change in angle of at least 45 or so degrees, typically more like 90, but for some reason putting the lower bound as high as 40 removed a lot of turns that seemed valid. This is likely the case for the same reason that we also had to cap the range at 170: the angles were very unpredictable and seemed inaccurate frequently. They often ended up being well over 180 and even over 360 which are not really reasonable at all for an ordinary car just driving down a road. There is also an issue with multiple turns being put in the same place, which may have something to do with the car visiting the same turns multiple times at different angles, but could also have something to do with multiple points with over 20 degrees of angle change occurring in a single turn (this would probably not happen if we set the lower threshold for a turn higher, but then we would sacrifice some valid turn classifications. All of this is good evidence we should have either used a

different classification method or cleaned/processed our data better, or both). All in all the turn classification is probably our most unsuccessful part of this project.

Figure 9.0 An example of a correct turn classification.

Figure 10.0 An example of a flaw in our turn classifier

Resulting KML Files:

Example of Path KML File

```
<Placemark id="3935">
        <name>Route </name>
        <styleUrl>#3936</styleUrl>
        <LineString id="3934">
           <coordinates>-77.54151333333333,43.13746,28.237</coordinates>
           <extrude>1</extrude>
           <altitudeMode>relativeToGround</altitudeMode>
        </LineString>
</Placemark>
```

Example of Stop KML Placemark File

```
<Placemark id="267">
        <name>Stop Light</name>
        <styleUrl>#268</styleUrl>
        <Point id="266">
           <coordinates>-77.485515,43.15349166666667,5.894321428571429</coordinates>
        </Point>
</Placemark>
```

Example of Left KML Turn

```
<Placemark id="275">
        <name>Left Turn</name>
        <styleUrl>#276</styleUrl>
        <Point id="274">
           <coordinates>-77.43775,43.138353333333335,43.138353333333335</coordinates>
        </Point>
  </Placemark>
```

Example of Right KML Turn

```
<Placemark id="287">
        <name>Right Turn</name>
        <styleUrl>#288</styleUrl>
        <Point id="286">
           <coordinates>-77.43774833333333,43.13839,43.13839</coordinates>
        </Point>
</Placemark>
```

Discussion

One issue we had when converting the GPS data to KML was removing any duplicated data points. As mentioned earlier, when the car comes to a stop, there will be multiple data points that

have the same location but the direction will vary. If we chose to leave this be and convert it to KML, the resulting path would be very sloppy and appear to be going all over the place, see **Figure 4.0**. We decided it would be best if we were to cluster these duplicated points to create a single data point for that location and that would remove any noise that was in the GPS data. We used K-means clustering of these duplicated data points to merge them all into a single point and the result viewed in Google Earth was much cleaner, see **Figure 4.0**. Since we used K-means clustering it caused some line strings to be placed all over as in **Figure 2.0**. So instead we decided to use line segments as the place markers in order to visualize the data properly. Another issue we faced was again determining when the car came to a stop. We chose to use DBScan method of classification of the data points to determine whether they were classified as a stop or not. We determine if points were within the radius epsilon that they were in fact data points where the car had come to a stop.

Conclusion

Overall, this project was interesting in actually being able to visualize how data mining techniques apply to a real world application such as processing GPS data. We have learned through this project how to convert GPS data files into multiple KML files that will be used on Google Earth or other KML processing software to visualize paths traveled by a car, Through this we have learned to properly preprocess data before doing any sort of cluster, classification or other methods of data mining. When creating the KML file we found there to be multiple duplicated data points of the same location. We decided to use both K-means clustering and agglomeration techniques to reduce the number of data points in order for the KML file to render

with less lag and to be visualized easier. Also, through this project we learned what method of clustering (K-Means or agglomeration) was better to use in certain situations. Determining a turn was one of the more difficult parts of this project. Deriving path angles from the GPS data proved to be harder than expected and resulted in frequent anomalies such as angles that would suggest Kinsman's car instantaneously spun all the way around (maybe it did?) and other similar issues. When anomalous data was recognized and accounted for the turn classifier became somewhat decent, allowing for a reasonable amount of turns to be caught accurately using only a simple decision stub checking for angles large enough to be considered a turn. Negative angles were assumed to be left turns and vice versa. Attempts were made to improve the classifier by considering speed and/or acceleration as well, but all thresholds tested for these ended up removing valid turn classifications and so this idea was eventually abandoned.

This particular project of processing GPS data can be very useful for logistics companies such as UPS, USPS, FedEx and more. Using data mining and machine learning techniques these companies can determine what ways to save the company the most money. For example, in order to save more money, UPS made it so their delivery trucks can only take right turns on deliveries in order to save on gas which will save the company more money. They figured this out using GPS processing and Shortest path algorithms to determine the path to take to a destination by only turning right. Another issue that can be solved is avoiding traffic on the highway or congested areas. A GPS processing software could look ahead of the current location and determine whether or not there is traffic and if so what route to take in place of the current. This is currently being done but is continuing to become more and more advanced as data mining and machine learning algorithms and techniques become more sophisticated.