Principles of Data Mining: GPS Project

Alvin Lin and Stefan Aleksic

August 2018 - December 2018

Background

What is GPS? What was the original intent? How many satellites are there? What other relevant details should we know about GPS? What else did you find out that is interesting to GPS?

Program Pattern

We wrote the program with Python, making generous use of the `numpy` library to handle the large quantities of data and the `fastkml` and `shapely` libraries to handle formatting the results of our data aggregation into a KML file for output. These libraries make use of Python's `lxml` module to handle the structuring of the XML necessary to follow create the KML file. Due to the efficiency of `numpy`, we were able to store the preprocessed input file entirely in memory for processing.

We did not use any particular design patterns, but separated out generic utility methods into a utility file (`util.py`). The utility file contained functions for parsing the longitude and latitude from the GGA and RMC format, as well as functions for calculating great circle distance and bearing between two latitude-longitude GPS coordintes.

The first thing we did with the input data was clean it. To do this, we calculated the distance between each consecutive point and pruned out consecutive points that were closer than a threshold. We determined this threshold through manual testing until we achieved an array of points that were close enough to resolve detail but sparse enough to remove clutter.

Detecting Left Turns

To detect left turns in the data, we calculated the bearing between each consecutive pair of latitude-longitude coordinates. We used a sliding window across this resulting array of bearings to calculate a signed delta in bearing across the window. If the delta was greater than 60 degrees, we determined that section of the window to be a left turn and labeled it as such.

We did not have to do any noise removal or signal processing since much of the noise was removed in the preprocessing stage where we cleaned out redundant points based on adjacent proximity. However, the sliding window method would tend to place many points across the entirety of a turn, so as soon as a left turn is found when the bearing delta goes over the threshold, we shift the window past the entire turn to avoid relabeling the turn. We also had problems with detecting turns that were prolonged over a longer stretch (such as the drive down Andrews Memorial Drive), which we handled by increasing the window size.

Detecting Stop Signs

Results

Running `project.py` will output the KML file to stdout instead of outputting the coordinates of the left turns and stop signs.

```
python project.py > output.kml
```

Importing the KML into Google Earth will show the following:

Summary and Conclusion

If you have any questions, comments, or concerns, please contact me at
alvin@omgimanerd.tech