TECHNOLOGY BRIEF

April 2021
Category: Common Vehicle Interface Initiative

Curve Logging

Summary

‘Curve Logging’ is a data sampling method that uses in-vehicle data buffer and edge processing to keep only the necessary data points to give the most accurate representation of the full data set as possible dictated by a predetermined allowable margin of error. This process provides accurate data and significantly reduces vehicle to cloud communication costs.

Today's vehicles generate large amounts of data that can be useful for a wide array of purposes from individual vehicle insights and fleet management to aggregated insights from many vehicles. If time based sampling of in-vehicle data is used, the higher the sampling frequency the closer the data sample is to the underlying parameter generating the data. However, a high frequency of data collection may result in largely redundant data sets and high data transmission costs from the vehicle to cloud

This tech brief describes the Curve Logging algorithm.

Key Characteristics

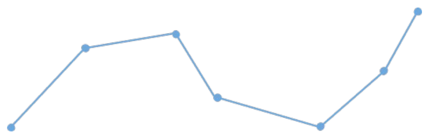
- In-vehicle edge processing of data to optimize transfers.
- Defines minimum data set still accurate within a set error limit.
- Simple algorithm -> possibility to implement with high performance.
- Applicable to many types of multi-dimensional data.

Description

Geotab’s curve logging algorithm is based on the Ramer-Douglas-Peucker algorithm. This allows for the simplification of a set of data to the least amount of required points to capture the data within an acceptable error value, the *curve error*.

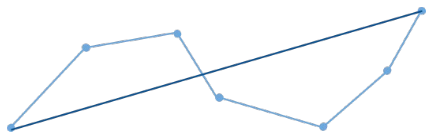
Step 0:

A set of data points (here represented by an x-y graph)



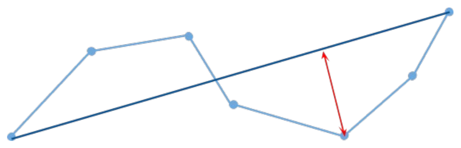
Step 1:

Start the curve algorithm by drawing a line from the first point to the last point.



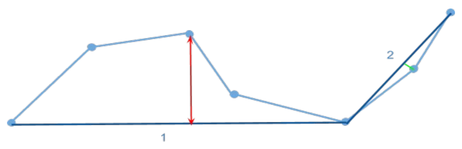
Step 2:

Find the point that has the largest perpendicular distance from this new line. If the largest distance is bigger than our allowed *curve error* then we need to keep that point. In this case it is:



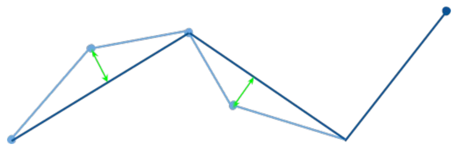
Step 3:

The largest distance point is kept and a line is drawn from the first point to this retained point and from the retained point to the last point. The curve algorithm is now run again on these two new lines. The point with the largest perpendicular distance from line 1 is bigger than our curve error (red line). The point with the largest perpendicular distance from line 2 is less than our curve error (green line). We keep the point at the end of the red line and discard the point at the end of the green line.



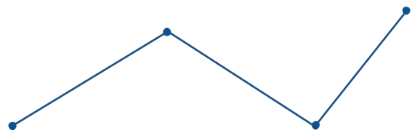
Step 4:

Get the largest perpendicular distances from the two new lines. Neither of them are larger than our curve error and so those points can be discarded.



Step 5:

The transferred representation of the data, still within the allowed *curve error*.



Distance measurement methods

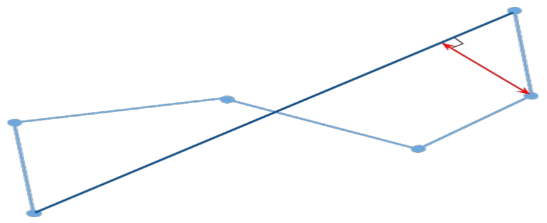
In the previous example a distance is calculated between the point which is considered and the line drawn from the first to last point. There are different methods of calculating this distance, and they each have pros and cons. The two methods used in our library are vertical and perpendicular distance.

Vertical distance

Vertical distance is a straight line up or down from the point that is considered to the line between the points already saved. The advantage of calculating the distance in this way is that the units of the distance will be the same units as the data which is collected. Making the results easier to interpret and it can be easy to know what error is introduced by using the curve logic, instead of keeping all the points.

Perpendicular distance

Another way distance can be to calculate is as a line perpendicular from the point that is considered, to the points already saved. Perpendicular distance is typically used for data which has the same units on both dimensions, such as GPS coordinates. Perpendicular distance can also be used for data which has different units on each dimension, for example speed data. When measuring the distance in this way, the units of the error will be a combination of the units the data is collected in, as well as the time units. This gives the developer more flexibility over which points are considered important, taking into account the time aspect as well. The disadvantage of using perpendicular error when units are different, is that more testing and tuning of parameters is required.



The public facing GitHub repository includes all the information required to implement 'curve logging'. This includes example applications and a description of error handling for the data sets. It also includes examples that enable comparison of time based data sampling methods with curve logging applied to sample data sets. Other examples provided on Github shows the application of curve logging to speed data leading to an 85% reduction in data, and application to GPS data leading to a 95% reduction in data.

We believe the power and benefit of curve logging is leveraged by its broad adoption. To this end, Geotab has opened up the required patent library to enable broad use within industry. We encourage all vehicle OEM's and Tier 1 suppliers to consider an implementation of open standard for curve logging to optimize data encoding wherever the algorithm applies to achieve faster data transfers and decreased bandwidth usage, while keeping data set accuracy compared to other solutions.

Alternatives & Related Technologies

Ramer-Douglas-Peucker algorithm
(https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm)

References & Additional Reading

Geotab's curve logging algorithm
(https://github.com/Geotab/curve)

Authors

Alex Sukov, Geotab
Ulf Bjorkengren, Geotab
Cristian Frincu, Geotab