Up-Stat 2016 Data Competition

Joseph S. Choi (The Institute of Optics, University of Rochester)

April 1, 2016

Problem Statement

Fill in missing data and determine algorithm to smooth the data points, generating 10 Hz data points.

Data Structure

Field	Description	Unit
GenTimeSec	Number of Seconds since Jan 1, 2004	Seconds
Latitude	GPS Latitude Position	deg
Longitude	GPS Longitude Position	deg
Elevation	GPS Elevation	m
Speed	GPS Estimated Speed	m/s
Heading	GPS Heading	deg
Ax	Estimated Longitudinal acceleration: acceleration in a straight line	m/s2
Ay	Estimated Lateral acceleration	m/s2
Az	Estimated Vertical acceleration	m/s2
Yawrate	Estimated Yaw rate: vehicle's angular velocity around its vertical axis	deg/s
ID	Mysterious string that contains id's of vehicles and transponders	Character

Theoretical Model

Algorithm Goals

From the laws of physics, the following goals must be achieved:

1. The position (**r** = (x,y,z)) of the car must be continuous as a function of time.
2. The velocity (**V** = (Vx,Vy,Vz)) and acceleration (**A** = (Ax,Ay,Az)) must be continuous as a function of time. This ensures that the position is smooth and continuous.

These are then the main goals of any algorithm for this problem.

Another goal would be:

3. Ensure that all data points are smooth and continuous as a function of time.

Deduced Position from Data

Using longitude and latitude, we can determine the position "x" and "y" using the Spherical Law of Cosines formula provided:

```
LatLon2Dist = function(Lon1, Lat1, Lon2, Lat2){
  point1.lat.rad = Lat1*pi/180
  point1.lon.rad = Lon1*pi/180
  point2.lat.rad = Lat2*pi/180
  point2.lon.rad = Lon2*pi/180
  d = acos(sin(point1.lat.rad)*sin(point2.lat.rad)+cos(point1.lat.rad)*cos(point2.lat.rad)*cos(
point2.lon.rad-point1.lon.rad))*6371000
    return(d)
}
```

Note that the above formula gives distances, so to obtain x(t)-x(t=0), we simply fix Latitude1=Latitude2, and vice versa for y(t)-y(t=0). For z, we simply use the Elevation. These give deduced (x,y,z) values from the data set, which I will call "$(x, y, z)_{measure}$."

Theoretical Position

It is important to compare the deduced $(x, y, z)_{measure}$ with a theoretical position vector $(x, y, z)_{theory}$. The theoretical position can be obtained by integration. For example,

$$x(t) = x(0) + Vx(0) * t + \int_0^t \int_0^{t'} Ax(t'')\, dt''dt'$$

and similarly for y(t) and z(t). This is because the **x, y, z** basis vectors are orthogonal, so we can neatly separate each component. The **A**(t), **V**(t), and **r**(t) given above are the theoretical values that our algorithm will optimize, and from which the final output data will be generated.

Parameters to Optimize

The parameters we wish to optimize and determine are:

1. Ax
2. Ay
3. Az
4. Vx
5. Vy
6. Longitude
7. Latitude
8. Elevation
9. Heading
10. Yawrate

The "speed" in the data table will be determined by Vx, Vy as follows:

$$\sqrt{Vx^2 + Vy^2}$$

Cost Function

To optimize, the algorithm will seek to minimize the following Cost Function J(ID), for *each* unique ID:

$$\begin{aligned} J(ID) = \frac{1}{2m} \sum_{t=\text{all time for ID}} \Big\{ & \theta_x \left(x_{theory}(t) - x_{measure}(t)\right)^2 + \theta_y \left(y_{theory}(t) - y_{measure}(t)\right)^2 \\ & + \theta_z \left(z_{theory}(t) - z_{measure}(t)\right)^2 + \theta_{Ax}(Ax(t) - Ax_{measure}(t))^2 \\ & + \theta_{Ay}(Ay(t) - Ay_{measure}(t))^2 + \theta_{Az}(Az(t) - Az_{measure}(t))^2 \\ & + \theta_{speed} \left(\sqrt{Vx^2(t) + Vy^2(t)} - speed_{measure}(t)\right)^2 \\ & + \theta_{H\&Y} \left(\frac{dHeading}{dt}(t) - Yawrate_{measure}(t)\right)^2 \\ & + \theta_{Heading}(Heading(t) - Heading_{measure}(t))^2 + \theta_{Yaw}(Yawrate(t) - Yaw_{measure}(t))^2 \\ & + TimeVariance(All\ Parameters) \Big\} \end{aligned}$$

where the "measure" values are that obtained from the actual data set. Also,

$$TimeVariance(All\ Parameters) = \left\{ \left(\frac{\Delta Ax}{\Delta t}\right)^2 + \cdots \right\}$$

Is the sum of the square of the time derivatives of ALL parameters being optimized. This is to satisfy the 3rd goal of ensuring smooth and continuous data generation.

The θ's above are determined from the Training data set.

Gradient Descent Optimization

To achieve optimal values for the 10 parameters, as a function of time, we use the gradient descent method.

Implementation

The following are the high level steps I have used to achieve

1. Aggregate data for each ID, to determine starting time and position. These will be set to t=0 and **r=0** for each ID, so as to make comparison easily between other ID's.
2. For each row and each ID, determine time from t=0, and position from (0,0,0).
3. Interpolate missing time data, since not all 10 Hz measurements obtained. For an initial interpolation, simple linear regression between points was used.
4. With initial interpolation, use Training Data to determine θ's in the Cost Function by summing cost function J(ID) for all ID's.
5. With the learned θ's now fixed, for each ID, minimize the cost function J(ID) by varying the 10 parameters. Iterate sufficiently to obtain convergence of J(ID).

6. For each ID, plot the parameters, (x,y,z) and other metrics to view smoothness and continuity.
7. Iterate steps 3-6 to improve further.

Additional Details

Assumptions

1. Although each instrument on a car and each transceiver may vary in characteristics, we ignore these to obtain a collective (or, average) model that characterizes the system of instruments as a whole. This is obtained from the training data and then generalized and applied to ALL instruments as if they were identical.
2. Data is NOT assumed to be time-ordered. If it IS time-ordered, then code can be modified to run faster.
3. Data for each ID are ALL contained in neighboring rows. So if rows 5-10 have ID=ID-A, and rows 4 or 11 contain different ID's, then only rows 5-10 contain data for ID-A.
4. Input times (GentimeSec) is always in increments of 0.1 sec exactly.
5. GPS speed only refers to the longitude and latitude speed, ignoring change in elevation.

Additional Algorithm Details

1. For each ID, determine origin, the starting coordinates in time and position:
    - Origin in Time = min(GentimeSec) => Set this to TimeSec=0
    - Origin in 3D Position (Longitude,Latitude,Elecation) = Position when GentimeSec == min(GentimeSec) => Set this to be our "(x,y,z)"=(0,0,0) origin.
2. To calculate the position "(x,y,z)" in units of length, we convert from the angles in (Longitude, Latitude, Elevation) as follows:
    - Use the Law of Cosine for 1 coordinate at a time. For example, to obtain "x", we fix Latitude ("y") and only obtain the distance using the Longitude change from the origin.
    - For "z", we simply use Elevation which is already in units of length.
3. May need to clean up data so 360 deg = 0 deg (heading, etc.)