Analyzing Street Bump Data - AnStreetBump

August 2011

Contents

List of Figures

1 Usage of AnStreetBump.py

AnStreetBump.py is written in python and its input file is a CSV file called track_input.csv. The code was developed and run on an x86-64 linux machine. The operating system is Opensuse 11.3.

Here is the format specification for track_input.csv:

1. Index - Any integer (not used by the program)
2. File_prefix - File_prefix+'.csv' is the file name for the StreetBump data.
3. Hole_accuracy_in_meters - This is the upper bound for the bin size of the pairs (Latitude,Longitude). This is used to report pot hole locations.
4. Holes_threshold - This number shall be referred as HT in the rest of this document. If the accelerometer data is HT standard deviations above the gravity vector of the device, the data point may indicate a pot hole.
5. Epsilon - A small positive real number. Any real number greater than -Epsilon is considered positive. Any real number less than +Epsilon is considered negative.
6. One_phone - If One_phone = 1, the data set contains data for a single device. Otherwise, the data set contains data for multiple devices.

1.1 Example 1

The first example uses SelfTest_newdata.csv.

Here is the track_input.csv :

```
Index,File_prefix,Hole_accuracy_in_meters,Holes_threshold,Epsilon,One_phone
19,SelfTest_newdata,10,2.9,0.2,0
```

To run AnStreetBump.py, make sure the code and all the input files are in your working directory. At the execution prompt: type

```
Python AnStreetBump.py
```

Here is the output on standard output for the Example 1:

```
More than one data set for same time stamp 434 435
Maximum time step :  90920
Minimum time step :  764
Average time step :  3351
No. of time step > 5000 : 12
The data is sorted.
A new file based on the sorted data has been created: sorted_SelfTest_newdata.csv
The input file is  sorted_SelfTest_newdata.csv
Phone  0
Gravity Vector : [-0.49331224  0.23059612  9.70038682]
Rput Matrix:
[[ 0.99458619  0.09195855  0.04839356]
 [-0.0906654   0.99547993 -0.02827518]
 [-0.05077496  0.02373448  0.99842805]]
Phone Orientation in car:  [ 5.27778222 -1.36001323  2.91126337]
No. of pot holes: 3
minimum and maximium time step of holes:  2.107 2.107
Number of Type 0 pot holes: 0
Number of Type 1 pot holes: 3
Phone  1
Gravity Vector : [-0.33833998  0.02361498  9.45607293]
Rput Matrix:
[[-0.98875761  0.14519296 -0.03574052]
 [-0.14518886 -0.98940021 -0.00272402]
 [-0.03575719  0.00249573  0.99935739]]
Phone Orientation in car:  [  1.71651520e+02  -1.42994935e-01   2.04917904e+00]
No. of pot holes: 2
minimum and maximium time step of holes:  1.269 4.219
Number of Type 0 pot holes: 0
Number of Type 1 pot holes: 2
Total number of pot holes : 5
The output file is  Solution_SelfTest_newdata.csv
```

Here is the content of file Solution_SelfTest_newdata.csv:

```
longitude,latitude,severity
-71.067794,42.374238,2
-71.067909,42.374245,3
-71.068037,42.374575,7
-71.071754,42.381241,3
-71.071823,42.381398,3
```

1.2 Example 2

Example 2 uses SelfTest.csv.

Here is the track_input.csv :

```
Index,File_prefix,Hole_accuracy_in_meters,Holes_threshold,Epsilon,One_phone
12,SelfTest,10,2.9,0.2,0
```

Here is the output on standard output for the Example 2:

```
Maximum time step :  1638824701
Minimum time step :  1
Average time step :  211704
No. of time step > 5000 : 880
The data is sorted.
A new file based on the sorted data has been created: sorted_SelfTest.csv
The input file is  sorted_SelfTest.csv
Phone  0
Gravity Vector : [-0.06597911  4.43508008  7.96061717]
Rput Matrix:
[[ 0.99982501 -0.01154571  0.01471917]
 [ 0.01724928  0.87350442 -0.48651052]
 [-0.00724015  0.48667928  0.87355072]]
Phone Orientation in car:  [ -0.75727366 -29.12255274   0.47486747]
No. of pot holes: 11
minimum and maximium time step of holes:  0.155 0.296
Number of Type 0 pot holes: 3
Number of Type 1 pot holes: 8
Phone  1
Gravity Vector : [ 0.5889474  -4.8744342   7.39763357]
```

```
Rput Matrix:
[[ 0.9881804  -0.07951084 -0.13106308]
 [ 0.13820103  0.83203126  0.53723781]
 [ 0.06633235 -0.54900093  0.83318546]]
Phone Orientation in car:  [ -5.45874144  33.29849928  -4.55188504]
No. of pot holes: 19
minimum and maximium time step of holes:  0.108 0.497
Number of Type 0 pot holes: 4
Number of Type 1 pot holes: 15
Phone  2
Gravity Vector : [-1.32541631  4.34392625  7.8662207 ]
Rput Matrix:
[[-0.98902466 -0.09103589 -0.11637307]
 [ 0.02318492 -0.87349846  0.4862745 ]
 [-0.14592013  0.47823938  0.86602218]]
Phone Orientation in car:  [ 185.94987814  -28.57047589    9.56419837]
No. of pot holes: 2
minimum and maximium time step of holes:  0.211 0.327
Number of Type 0 pot holes: 1
Number of Type 1 pot holes: 1
Phone  3
Gravity Vector : [-0.65202944 -4.48706082  7.37933113]
Rput Matrix:
[[-0.24260797  0.83828201  0.48828746]
 [-0.96719896 -0.16994599 -0.18879762]
 [-0.07528316 -0.51807493  0.85201574]]
Phone Orientation in car:  [ 101.46032522   31.20320973    5.04947725]
No. of pot holes: 1
minimum and maximium time step of holes:  0.157 0.217
Number of Type 0 pot holes: 0
Number of Type 1 pot holes: 1
Total number of pot holes : 33
The output file is  Solution_SelfTest.csv
```

Here is the content of file Solution_SelfTest.csv:

```
longitude,latitude,severity
-71.068481,42.357738,4
```

```
-71.111069,42.350300,3
-71.112198,42.350754,5
-71.122475,42.308746,4
-71.122696,42.308403,3
-71.122734,42.308353,3
-71.123566,42.307205,4
-71.123726,42.306999,2
-71.123825,42.306919,4
-71.144928,42.258183,4
-71.144936,42.258141,4
-71.144905,42.258131,3
-71.144836,42.257019,3
-71.144798,42.256971,4
-71.144814,42.256973,4
-71.144463,42.256531,4
-71.143799,42.265656,3
-71.145287,42.260071,4
-71.145233,42.260075,5
-71.145248,42.259945,4
-71.145035,42.258598,3
-71.143890,42.272831,3
-71.144310,42.272476,3
-71.124863,42.304173,4
-71.124847,42.304314,5
-71.124748,42.305164,4
-71.124794,42.304478,5
-71.124763,42.304539,4
-71.125458,42.302868,5
-71.127968,42.300282,3
```

2 Introduction

This article describes an algorithm for the processing of output from the Street Bump program in response to InnoCentive Challenge 9932752. The goal of the algorithm is to predict the location of pot holes from the output of the Street program. The code is written in Python.

The algorithm has stages:

- Read the input file (output from Street Bump Program), sort data with time stamps, eliminate duplicate records.
- Determine the number of devices and the data belonging to each device.
- Sort GPS locations.
- for each device :
  - Generate gravity vector and rotation matrix for the car.
  - Generate standard deviation for the accelerometer data.
  - Identify pot hole locations and severity
- Report pot holes from all the phones

In the rest of this article, we shall describe each steps in detail, including the assumptions made. We shall discuss the limitation of the algorithm and propose future improvement.

3 Reading and Sorting Input Data

The input data is in csv format. However, in some cases, the time stamps are not in chronological order, and some of the records are duplicated. We sort the input data on the time stamps and eliminate duplicate records. Then we rewrite the input file with the file name of the form "sorted_fname.csv" where "fname.csv" is the file name of the original input data set. All processing is done with the new file "sorted_fname.csv".

If the time stamps are in chronological order and there are no duplicated records, no new file would be generated.

4 Multiple Devices

The bearing and orientation data are used to determine the the number of devices and the data belonging to each device. The triplets $(OrientX - Bearing, OrientY, OrientZ)$ for the data points are put into bins by the cube-tree algorithm. An upper bound for the bin size is $\pi/2$

The most populated bin and all the bins adjacent to it are picked. All the data contained in these bins are picked as data for the first device. The process is repeated for the rest of the bins until all the bins are picked.

This method is heuristic. However the method can differentiate devices with their $z-$ axes pointing at different directions.

5 Sort GPS Locations

The pair $(Latitude, Longitude)$ for the data points are put into bins by the quad-tree algorithm. The upper bound for the size of the bin is determined by the input parameter "hole_accuracy_in_meters".

These bins are used in reporting pot holes. If more than one pot hole occurs in one bin, the mean of these pot holes is reported as a single pot hole in the bin.

6 Gravity Vector, Rotation Matrix of the Car and Standard Deviation

We define the gravity vector as the weighted average of the accelerometer vectors $(A_{x_i}, A_{y_i}, A_{z_i}), i = 0, 1, \cdots, N-1$, where N is the total number of unique time stamps:

$$(g_x, g_y, g_z) = \sum_{i=1}^{i=N-1} \frac{1}{2DT}[(A_{x_{i-1}}, A_{y_{i-1}}, A_{z_{i-1}}) + (A_{x_i}, A_{y_i}, A_{z_i})]dt_{i-1}.$$

where $dt_i = t_{i+1} - ti$, and $DT = t_{N-1} - t_0$.

To discuss our method for computing the rotation matrix of the car, we need some background discussion concerning coordinates systems. There are three standard coordinate systems:

1. The phone coordinate system as described in the Android System documentation shown in Figure 1.
2. The world coordinate system where y is tangential to the ground and points to the magnetic north and z points to the sky and is perpendicular to the ground and $x = y \times z$ as shown in Figure 2.
3. The true north coordinate system where y is tangential to the ground and points to the true north and z points to the sky and is perpendicular to the ground and $x = y \times z$.

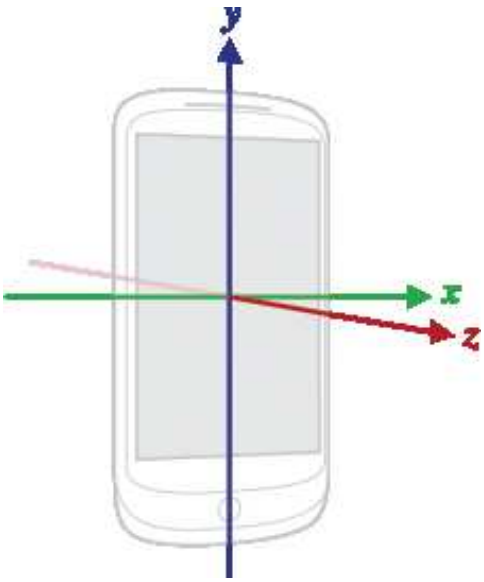

Figure 1: Phone Coordinates

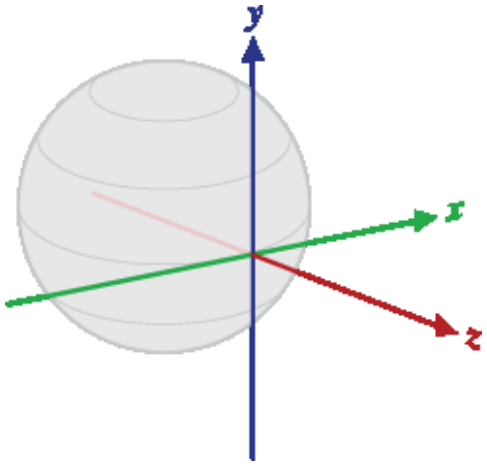

Figure 2: World Coordinates

We define a fourth coordinate system, the car coordinate system, where x is tangential to the ground and points to the direction of travel, z points to the sky and $y = z \times x$. We define the rotation matrix R_{put} as the rotation matrix that transforms a vector in phone coordinates to car coordinates.

For example, if the phone is placed in the car in a cup holder, facing up, sat perpendicular to the direction of travel, with the top on the driver's

side, as in Training Data Sets 1 to 4, the R_{put} matrix should be the identity matrix.

To compute the R_{put} matrix, we use the gravity vector, the magnetometer data, and the bearing data and the fact that at Boston, the magnetic north is 15 degrees west of the true north.

Let v_{phone}, v_{car}, v_{world}, and v_{north} denote vectors of the phone, the car, the world and the true-north coordinate system. Then $R_{put}v_{phone} = v_{car}$.

Let R be the rotation matrix based on the accelerometer and magnetometer data. Then $Rv_{phone} = v_{world}$. Let R_b be the matrix related to the bearing. Then $R_b v_{car} = v_{north}$. Let R_s be the matrix that maps the magnetic north to the true north for Boston. Then $R_s^{-1}v_{north} = v_{world}$. Combining all these equations with substitution, we obtain the following equation:

$$Rv_{phone} = v_{world} = R_s^{-1}v_{north} = R_s^{-1}R_b v_{car} = R_s^{-1}R_b R_{put}v_{phone}$$

Then $R_{put} = R_b^{-1}R_s R$, because v_{phone} is any arbitrary vector in phone coordinates.

Note that R_b^{-1} and R_s are rotation matrix around the z axes in world coordinates, therefore they commute with each other. The R_{put} equation can be written as

$$R_{put} = R_s R_b^{-1} R$$

We ported two subroutines from the Android Sensor Manager: getRotationMatrix and getOrientation to support this calculation. Further details are presented in the Appendix.

R_{put} reflects how the phone is placed inside the car. Among the 12 training data set, the R_{put} matrices computed for 10 cases are consistant with the phone placements provided.

7 Pot Holes Identifications

We define D_{z_i} as the number of standard deviations from the gravity and L_{z_i} as the linear acceleration in the z component in car coordinates. Note that $|L_{z_i}| = D_{z_i}$. We are only interested in the cases where $D_{z_i} > HT$.

7.1 Type 0 Pot Hole

We assume when the car hits a pot hole, the car would go down first then go up. It either would go down abruptly or go up abruptly.

We consider $L_{z_i} > 0$ and $L_{z_i} < 0$, separately:

1. In the case $L_{z_i} > 0$, if $L_{z_{i-1}} < 0$ and $L_{z_{i-2}} > -\epsilon$ and $L_{z_{i+1}} < HT$, add the pair of time stamps : $(i-1, i)$ to the pot hole time stamp list.

2. In the case $L_{z_i} < 0$, if $L_{z_{i+1}} > 0$ and $L_{z_{i+2}} < \epsilon$ and $L_{z_{i-1}} > -HT$, add the pair of time stamps : $(i, i+1)$ to the pot hole time stamp list.

The following figure illustrates the algorithm described above.

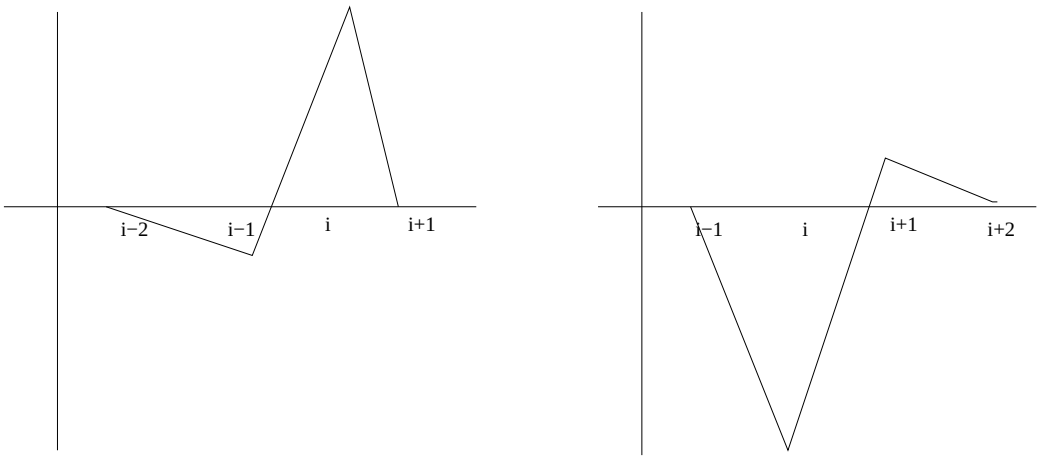

Figure 3: Time Stamps v.s. Linear Acceleration in z Direction

We call this the "Down-Up" algorithm, and the pot holes identified by the algoritm "Type 0" pot holes.

We note GPS data is updated on the average every second, and the accelerometer data for Type 0 pot holes are updated about 5 times more frequently than the GPS data. For a Type 0 pot hole, we report the location of the pot hole at time stamp $i + k$ where k is the smallest positive integer such that $\sum_{m=i}^{k} dt_m \geq 1sec$.

7.2 Type 1 Pot Holes

We assume the car's experience of a pot hole to be around .25 seconds. The "Down-Up" algorithm is only applicable for time steps less than or around .25 seconds. When the time stamp for $D_{z_i} > HT$ is greater than 1 second, we simply report the car hits a pot hole at $Time_i$. We call this a "Type 1" pot hole.

We added the Type 1 pot hole after studying the new data sets from version 1.3 of the StreetBump program, in particular, SelfTest_newdata.csv. In these data sets, most of the time steps for $D_{z_i} > HT$ are larger than one second. Yet with the 13 data sets from version 1.2 of the StreetBump program, we can apply the "Down-Up" algorithm.

7.3 Severity

A Type 0 pot hole is identified by a pair of time stamps $(k, k+1)$. The severity is defined as $D_{z_k} + D_{z_{k+1}}$. A Type 1 pot hole only has one time stamp k, and the severity is defined as D_{z_k}

7.4 Reporting Location

We note the data are collected from multiple devices with the car driven in a loop more than once. It is quite likely that the same pot hole can be reported several times.

For concise reporting, we report one pot hole per bin (see Section 4). After we collect the pot holes data from all the devices, we also find the bins in which the locations lie. If there are more than one pot hole identified in a bin, we report the average.

8 Conclusions and Proposal of Future Work

We have presented an algorithm to identify multiple devices and pot holes based on the linear acceleration vectors in the z direction in car coordinates. The algorithms is based on assumptions

1. The car hit a pot hole when the $D_z > HT$ and the car would go down and then up abruptly.

2. There is about a one second delay in the GPS position.

3. To determine which are the data points associated with a single device, we assume the triplet (Azimuth - Bearing, pitch and roll) to be constant for a single device.

The algorithms generates both false positives and false negatives when applied to the training data sets. When the car hits a pot hole, it experiences a roll, but we could not see this roll in the training data sets. Also the assumption on the motion of the car going down then up abruptly only makes sense if the time step is less than 1 second when $D_z > HT$.

Our second assumption about GPS position is based on observation only. In theory, integrating the accelerometer data twice should yield displacement. But the accelerometer data appears to have too much noise for numerical integration to yield useful results.

As for the last assumption we observe the pitch and roll data permits identification with devices, but (Azimuth-Bearing) has a lot of noise. This could be due to the fact that the magnetometer of a phone placed in a car does not yield accurate compass data.

We suggest research to be done in the following areas:

1. Obtain more reliable orientation data. According to the latest Android document,Sensor.TYPE_ORIENTATION is deprecated.

2. Appropriate filtering of the accelerometer data to obtain accurate displacement information.

3. The Street Bump program output 30 seconds of calibration data when the car is standing still so that gravity and phone orientation information can be obtained.

4. Collection of data around anomalies should be more frequent - like version 1.2.

Appendix

The Computation of R_{put}

To calculate R_{put}, we use the following property of the matrix R.

R is a function of the gravity and magnetometer data as illustrated in the subroutine getRotationMatrix in the Android Sensor Manager. R can also be expressed as the product of three rotation matrices as illustrated in the subroutine getOrientation in the Android Sensor Manager.

$$R = R(\vec{E}, \vec{A}) = R_z(O_0)R_x(O_1)R_y(O_2)$$

$\vec{E}$ and $\vec{A}$ are respectively the magnetometer vectors and accelerometer vectors. (O_0, O_1, O_2) are respectively the azimuth,pitch and roll angles (in radians) described in "StreetBump CSV Format Specifications".

Note:

1. R_b is a rotation matrix $R_z(b)$ where b is the bearing.
2. Boston's magnetic north is 15 degrees from the true north. Therefore $R_s = R_z(75^\circ)$
3. For each time stamp i, we have a different magnetometer vector $\vec{E}_i$ and a different bearing angle, therefore a different R matrix, say R_i and a different R_b matrix, say R_{b_i}.
4. O_1 and O_2 are determined by the gravity vector, therefore. They are the same for all time stamps. This is not the case for O_0 which depends on both the gravity vector and the magnetometer vector.
5. $(R_z(b_i))^{-1} = R_z(-b_i)$.
6. We can write $R_{b_i}R_i$ as

   $$R_{b_i}^{-1}R_i = (R_z(b_i))^{-1}R_z(O_{0_i})R_x(O_1)R_y(O_2) = R_z(-b_i+O_{0_i})R_x(O_1)R_y(O_2)$$

Since R_{put} reflects how the phone is place inside the car, it should be fixed which implies $(-b_i + O_{0_i})$ should be constant if there were no measurement errors. Since this is not the case, we try to get the average of $(-b_i + O_{0_i})$.

To avoid branch cut problem, we compute sa, the average of $\sin(-b_i+O_{0_i})$ and ca, the average of $\cos(-b_i + O_{0_i})$, the $O_0 = \arctan(sa/ca)$.

Then

$$R_{put} = R_z(75^\circ + O_0)R_x(O_1)R_y(O_2).$$