Snap2Grid Project

Andrew Stevenson

PSU CS510 Summer 2020 Final Project

Breadcrumb Records

- C-Tran is the regional transit organization for Clark County in the State of Washington
- Monitors vehicles with a sensor network that collects GPS coordinates every 5 seconds
- Yields nearly 500K sensor readings (known as *bread crumb* readings) every day

Breadcrumb Records

	EVENT_NO_TRIP	OPD_DATE	VEHICLE_ID	METERS	ACT_TIME	GPS_LONGITUDE	GPS_LATITUDE
0	153067037	09-MAR-20	1776	19	18173	-122.60246	45.63782
1	153067037	09-MAR-20	1776	32	18178	-122.60260	45.63774
2	153067037	09-MAR-20	1776	51	18183	-122.60286	45.63777
3	153067037	09-MAR-20	1776	80	18188	-122.60324	45.63776
4	153067037	09-MAR-20	1776	111	18193	-122.60364	45.63775
5	153067037	09-MAR-20	1776	139	18198	-122.60401	45.63779
6	153067037	09-MAR-20	1776	164	18203	-122.60433	45.63783
7	153067037	09-MAR-20	1776	187	18208	-122.60463	45.63782
8	153067037	09-MAR-20	1776	197	18212	-122.60477	45.63782
9	153067037	09-MAR-20	1776	212	18217	-122.60494	45.63778
10	153067037	09-MAR-20	1776	246	18222	-122.60497	45.63747
11	153067037	09-MAR-20	1776	292	18227	-122.60495	45.63705
12	153067037	09-MAR-20	1776	313	18232	-122.60495	45.63683
13	153067037	09-MAR-20	1776	326	18237	-122.60495	45.63672
14	153067037	09-MAR-20	1776	353	18242	-122.60491	45.63648
15	153067037	09-MAR-20	1776	383	18247	-122.60456	45.63638
16	153067037	09-MAR-20	1776	430	18252	-122.60398	45.63640
17	153067037	09-MAR-20	1776	494	18257	-122.60316	45.63643
18	153067037	09-MAR-20	1776	567	18262	-122.60222	45.63647

GTFS Data

- GTFS is General Transit Feed Specification
- Google’s format for public transit geographic information
- Contains GPS coordinates of the intended bus route
- Note the difference between the planned route and the recorded trip—may differ

GTFS Records

	route_index	shape_index	shape_pt_sequence	shape_pt_lat	shape_pt_lon	shape_dist_traveled	route_short_name	route_long_name
2	2	2	0	45.61947	-122.63893	0.00000	6	6 - Fruit Valley / Grand
2	2	2	1	45.61951	-122.63894	4.16402	6	6 - Fruit Valley / Grand
2	2	2	2	45.61911	-122.64093	165.19724	6	6 - Fruit Valley / Grand
2	2	2	3	45.61898	-122.64158	217.47816	6	6 - Fruit Valley / Grand
2	2	2	4	45.61894	-122.64176	232.73356	6	6 - Fruit Valley / Grand
2	2	2	5	45.61892	-122.64193	245.48350	6	6 - Fruit Valley / Grand
2	2	2	6	45.61890	-122.64216	263.65265	6	6 - Fruit Valley / Grand
2	2	2	7	45.61886	-122.64294	324.45695	6	6 - Fruit Valley / Grand
2	2	2	8	45.61885	-122.64339	359.76361	6	6 - Fruit Valley / Grand
2	2	2	9	45.61966	-122.64338	449.34154	6	6 - Fruit Valley / Grand
2	2	2	10	45.62074	-122.64336	569.04170	6	6 - Fruit Valley / Grand
2	2	2	11	45.62088	-122.64336	584.48917	6	6 - Fruit Valley / Grand
2	2	2	14	45.62108	-122.64336	608.79914	6	6 - Fruit Valley / Grand
2	2	2	15	45.62186	-122.64336	695.37172	6	6 - Fruit Valley / Grand
2	2	2	16	45.62202	-122.64336	713.48629	6	6 - Fruit Valley / Grand
2	2	2	19	45.62217	-122.64336	737.69748	6	6 - Fruit Valley / Grand
2	2	2	20	45.62379	-122.64335	918.03439	6	6 - Fruit Valley / Grand
2	2	2	21	45.62384	-122.64330	924.81566	6	6 - Fruit Valley / Grand
2	2	2	22	45.62386	-122.64322	931.53825	6	6 - Fruit Valley / Grand

Project Goal, Part 1

- Project each breadcrumb to the nearest point on the planned route
- The naïve projections may have the crumbs out of sequence
- i.e. they may show the bus moving backward
- These points must be corrected so all projected points are in chronological order

Challenge 1:

- We have no way to match trips to shapes.
- We can match any given trip to several (2-20) shapes.
- So then just look at the average start and ending coordinates and find closest shape
- Build a linking table matching planned trips to estimated route
- Works for about 7 mil, of 8.5 mil records total

Challenge 2:

- The shapes had little off-road jogs in them at every bus stop
- Associated with duplicate coordinates before and after the jog
- Bruce’s recommended solution was just to delete them as they interfere with our projections

Challenge 3:

- 8.5 million records need to be processed
- Large enough quantity that I have to be careful with the algorithm
- A lot of time learning pandas in enough depth to process these records efficiently
- Wound up making my own pipeline, breaking dataset into small files and using several processes running simultaneously

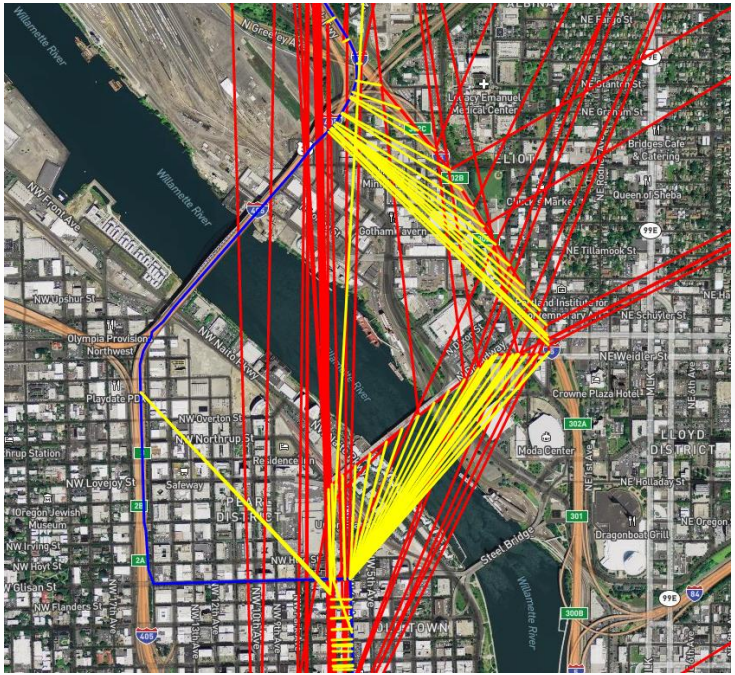
Project Goal, Part 2

- Second task was to compute a trip deviation metric
- Purpose of metric is to sort trips based on the likelihood that the driver actually took a different road than assigned
- As opposed to just noise in the data

Suspicion Level Metric

- Pick an arbitrary threshold, say 60m (the length of a PDX city block.)
- For each trip, find the percentage of crumbs with deviation value above the threshold
- This percentage is the suspicion score
- Sort by this score and inspect the records at the top of the list first

Results

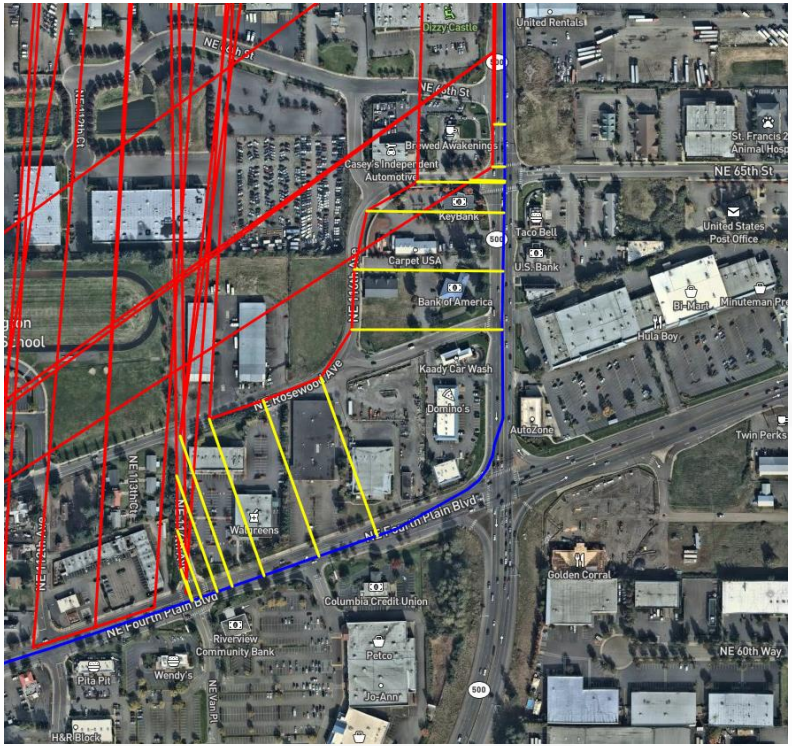


Blue: Planned Trip

Red: Recorded trip (all over the place)

Yellow: Deviation measurements over 60 m

Results



Blue: Planned Trip
Red: Recorded trip (all over the place)
Yellow: Deviation measurements over 60 m

Limitations/Further Development

- Fundamentally deviation values are not signed—meaning it's not possible to track 'runs' of large numbers of crumbs with deviations in the same direction
- In the future a web tool would be very convenient for viewing the contents of the crumbs data

Thanks for Listening

- Thanks to Bruce Irvin for his assistance
- Thanks to other working group members
- Questions?

Appendix A: Project Goals

Part A: Find Trips with Most/Least Deviation

If we can find the trips with the most deviation, then we will have given information to C-Tran that (a) they do not currently have and (b) they would find valuable, either to learn of cases in which drivers purposely choose a different path than the planned route or cases in which a vehicle's GPS sensor needs servicing.

Part A could/should be broken into sub-goals.

- develop a way to join the breadcrumb readings with the geometry (shapes.txt) of the planned trips.
- develop an algorithm for computing single point deviation given a breadcrumb sensor reading and the geometry of a planned trip.
- develop a method for combining individual point deviations into a "trip deviation" metric. This metric should be scalable (or normalized) such that long trips can be compared with short trips.
- compute the trip deviation metric for all trips in the sample data.
- sort and present the results.
- [STRETCH GOAL] provide a way to visualize recorded trips and planned trips so that we can visually inspect the deviations. Possibly use Sam Gomena's tool to do this or develop a separate LeafletJS-based tool for this.

Appendix B: Deviation Algorithm

COMPUTE-DEVIATIONS(S, B)

```
1  Sort S by pointOrder
2  for every record r_i ∈ S
3      if r_i.pointOrder ≠ 0
4          Set r_i.distTraveled = dist(r_i, r_{i−1})
5      else
6          Set r_i.distTraveled = 0
7  for every record r ∈ B
8      r.projection = nearest point on all segments in S
9      r.distTraveled = r.projection's distance along S
10 Sort B by tripID, time // Note we don't compare crumb records to each other until this point
11 for Each unique tripID t ∈ B
12     for Each record r_i ∈ B such that r.tripID = t
13         if r_i.distTraveled < r_{i−1}.distTraveled
14             Look backward in t to find correct projection halfway between new neighbors
15         elseif r_i.distTraveled > r_{i+1}.distTraveled
16             Look forward to find correct projection halfway between new neighbors
17 return B
```

Appendix C: Sample Output

	tripID	timestamp	vehicleID	origLatitude	origLongitude	shapeID	routeID	plannedTripID	correctedLatitude	correctedLongitude	distance	angle	suspicionLevel
2343446	153377137.00000	1584049054.00000	2268.00000	45.62872	-122.66795	11	6	1399.00000	45.62873	-122.66795	1.34513	0	0.00738
2343447	153377137.00000	1584049049.00000	2268.00000	45.62873	-122.66865	11	6	1399.00000	45.62875	-122.66865	2.49448	0	0.00738
2343448	153377137.00000	1584049044.00000	2268.00000	45.62873	-122.66925	11	6	1399.00000	45.62876	-122.66925	3.20968	0	0.00738
2343449	153377137.00000	1584048894.00000	2268.00000	45.62873	-122.66986	11	6	1399.00000	45.62876	-122.66986	3.67065	0	0.00738
2343450	153377137.00000	1584048889.00000	2268.00000	45.62874	-122.67018	11	6	1399.00000	45.62876	-122.67018	2.48069	0	0.00738
2343451	153377137.00000	1584048884.00000	2268.00000	45.62875	-122.67074	11	6	1399.00000	45.62877	-122.67074	2.00956	0	0.00738
2343452	153377137.00000	1584048874.00000	2268.00000	45.62876	-122.67198	11	6	1399.00000	45.62878	-122.67198	2.61191	0	0.00738
2343453	153377137.00000	1584048719.00000	2268.00000	45.63207	-122.67711	11	6	1399.00000	45.63207	-122.67707	3.06217	0	0.00738
2343454	153940769.00000	1584651052.00000	6007.00000	45.64158	-122.61492	35	13	1851.00000	45.64166	-122.61496	8.71941	0	0.00738
2343455	153940769.00000	1584651062.00000	6007.00000	45.64197	-122.61342	35	13	1851.00000	45.64204	-122.61345	7.97282	0	0.00738
2343456	153940769.00000	1584651487.00000	6007.00000	45.65089	-122.58371	35	13	1851.00000	45.65089	-122.58370	1.18510	0	0.00738
2343457	153940769.00000	1584651432.00000	6007.00000	45.65086	-122.58377	35	13	1851.00000	45.65085	-122.58376	1.52781	0	0.00738
2343458	153940769.00000	1584651427.00000	6007.00000	45.65075	-122.58394	35	13	1851.00000	45.65074	-122.58393	1.90170	0	0.00738
2343459	153940769.00000	1584651417.00000	6007.00000	45.65022	-122.58487	35	13	1851.00000	45.65024	-122.58490	3.71667	0	0.00738
2343460	153940769.00000	1584651412.00000	6007.00000	45.65007	-122.58518	35	13	1851.00000	45.65012	-122.58522	6.13187	0	0.00738

Appendix D: Output Format Specification

Each row of the file represents a breadcrumb reading and has several columns as follows:

- **tripID** - the ID of the recorded trip
- **timestamp** - the moment at which the reading was taken
- **vehicleID** - identifies of the vehicle
- **origLatitude** - the original sensor latitude value
- **origLongitude** - the original sensor longitude value
- **shapeID** - the shape in shapes.txt that corresponds to tripID
- **routeID** - the route in route.txt corresponding to tripID
- **plannedTripID** - the planned trip corresponding to tripID (this value is not always correct and probably should not be used in your analyis)
- **correctedLatitude** - the corrected vehicle position
- **correctedLongitude** - the corrected vehicle position
- **distance** - the Euclidean distance (in meters) from the original sensor reading and the corrected vehicle position
- **angle** - this value is not implemented.
- **suspicionLevel** – the percentage of a trip's records having distance values above 60

Finis