us-101

In [40]:
```
import pandas as pd
import numpy as np
```

In [41]:
```
PATH_LOAD = "us-101.csv"
df = pd.read_csv(PATH_LOAD)
print(df.shape)
```

```
(4802933, 9)
```

In [48]:
```
df.iloc[5:10, :] # A section of dataframe
```

Out[48]:

	Unnamed: 0	Vehicle_ID	Global_Time	Real_Time	Local_X	Local_Y	v_Vel	Lane_ID	Movement
5	5	2	1118846980200	2005-06-15 10:49:40.2	16.467	35.381	40.0	2	NaN
6	6	5	1118846980200	2005-06-15 10:49:40.2	39.685	59.154	40.0	4	NaN
7	7	5	1118846980300	2005-06-15 10:49:40.3	39.665	63.154	40.0	4	NaN
8	8	2	1118846980300	2005-06-15 10:49:40.3	16.447	39.381	40.0	2	NaN
9	9	2	1118846980400	2005-06-15 10:49:40.4	16.426	43.381	40.0	2	NaN

In [44]:
```
class Vehicle:

    """
    This is a standard class for each vehicle.
    Variables:
    vID        is Vehicle_ID  in the processed datasheet.
    globalTime is Global_Time  in the processed datasheet.
    xLoc       is Local_X     in the processed datasheet.
    yLoc       is Local_Y     in the processed datasheet.
    vVel       is v_Vel       in the processed datasheet.
    laneID     is Lane_ID     in the processed datasheet.
    move       is Movement    in the processed datasheet.
    transRange is a pre-set parameter for each vehicle.
    haveData   is to indicate if this vehicle has the data or not.

    Main methods:
    update method is to feed real time data to this vehicle and update all its variables.
    """

    def __init__(self, vID, globalTime, xLoc, yLoc, vVel, laneID, move = float('nan'), transRange = 500, haveData = 0):
        self.vID = vID
        self.globalTime = globalTime
        self.startTime = globalTime # Initialize the startTime for each vehicle by the first globalTime
        self.lastTime = globalTime # Initialize the lastTime for each vehicle by the first globalTime
        self.xLoc = xLoc
        self.yLoc = yLoc
        self.vVel = vVel
        self.laneID = laneID
        self.move = move
        self.transRange = transRange
        self.haveData = haveData
        self.onRoad = onRoad


    def update(self, vID, globalTime, xLoc, yLoc, vVel, laneID, move, currentVehiclesList):
        if vID == self.vID: # Check if the feed data is for the vehicle itself
            self.globalTime = globalTime
            self.xLoc = xLoc
            self.yLoc = yLoc
            self.vVel = vVel
            self.laneID = laneID
            self.move = move
            if checkOnRoad():
                for vehicle in currentVehiclesList:
                    if checkVehiclesInRange(vehicle):
                        vehicle.haveData = 1
            else:
                self.haveData = 0 # Re-initialize the haveData in order to make this vehicle as a new vehicle
                self.startTime = globalTime # Re-initialize the startTime for this vehicle by new globalTime
                self.lastTime = globalTime # Re-initialize the lastTime for this vehicle by new globalTime




    def checkVehiclesInRange(self, other):
        """
        This method is to check if the selected vehicle is within the transmission range.
        """
        distance = np.sqrt( (other.xLoc - self.xLoc)**2 + (other.yLoc - self.yLoc)**2 )
        if distance < self.transRange:
            return True
        else:
            return False

    def checkOnRoad(self):
        """
        This method is to check if this vehicle is still on the road.
        """
        if self.globalTime == self.lastTime + 100:
            return True
        else:
            return False
```

In [45]:
```python
def coveragePercent(currentVehicleList):
    """
    This method is to calculate the percentage of vehicles that have the data in real-time.
    """
    countVehiclesHaveData = 0
    for vehicle in currentVehicleList:
        if vehicle.haveData == 1:
            countVehiclesHaveData += 1
    return countVehiclesHaveData/len(currentVehicleList)
```

Comments

As we can see from the report "Data_Analysis_20180930.pdf", there exists 3 tracks for the vehicle whose ID = 2. And we could find that its timeline is not continous. Thus we could use this point to write the method checkOnRoad to distinguish the old vehicle and new vehicle even their IDs are the same. The new vehicle will be re-initialize with new data and the variable haveData will be reset to original.

In [47]:
```python
dfID2 = df[df['Vehicle_ID'] == 2]
dfID2.iloc[435:440, :]
```

Out[47]:

	Unnamed: 0	Vehicle_ID	Global_Time	Real_Time	Local_X	Local_Y	v_Vel	Lane_ID	Movement
26298	26298	2	1118847023700	2005-06-15 10:50:23.7	8.416	2123.121	70.02	1	NaN
26409	26409	2	1118847023800	2005-06-15 10:50:23.8	8.410	2130.121	70.02	1	NaN
1133001	1133001	2	1118847864800	2005-06-15 11:04:24.8	38.844	84.876	14.98	4	NaN
1133112	1133112	2	1118847864900	2005-06-15 11:04:24.9	38.836	86.376	14.98	4	NaN
1133182	1133182	2	1118847865000	2005-06-15 11:04:25.0	38.828	87.875	14.98	4	NaN