In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [4]:
```python
PATH_LOAD = "Next_Generation_Simulation__NGSIM__Vehicle_Trajectories_and_Supporting_Data.csv"
df = pd.read_csv(PATH_LOAD)
```

Understand the Data - Overall

In [13]:
```python
# How many vehicles this spreadsheet has recorded
print('This spreadsheet contains', len(df.groupby(['Vehicle_ID'])), 'vehicles in total.')

# Filter out motorbike and truck
dfCar = df[df['v_Class'] == 2]

# How many cars this spreadsheet has recorded
print('This spreadsheet contains', len(dfCar.groupby(['Vehicle_ID'])), 'cars.')

# What is the max/min number of records for a single car
print('Max number of records for a single car is:')
print(dfCar.groupby(['Vehicle_ID']).size().max())
print('Min number of records for a single car is:')
print(dfCar.groupby(['Vehicle_ID']).size().min())
```

```
This spreadsheet contains 3233 vehicles in total.
This spreadsheet contains 3216 cars.
Max number of records for a single car is:
9834
Min number of records for a single car is:
264
```

In [19]:
```python
# How many roads this spreadsheet havs recorded
locationSeries = dfCar['Location']
locationList = []
for item in locationSeries:
    if item not in locationList:
        locationList.append(item)

print('This spreadsheet contains', len(dfCar.groupby(['Location'])), 'roads. They are:')
print(locationList)
```

```
This spreadsheet contains 4 roads. They are:
['us-101', 'lankershim', 'peachtree', 'i-80']
```

In [20]:
```python
# How many cars this spreadsheet have recorded for each road
df101 = dfCar[dfCar['Location'] == locationList[0]]
print('There are', len(df101.groupby(['Vehicle_ID']).size()), 'cars at ' + locationList[0] + ' .')

dfLan = dfCar[dfCar['Location'] == locationList[1]]
print('There are', len(dfLan.groupby(['Vehicle_ID']).size()), 'cars at ' + locationList[1] + ' .')

dfPea = dfCar[dfCar['Location'] == locationList[2]]
print('There are', len(dfPea.groupby(['Vehicle_ID']).size()), 'cars at ' + locationList[2] + ' .')

df80 = dfCar[dfCar['Location'] == locationList[3]]
print('There are', len(df80.groupby(['Vehicle_ID']).size()), 'cars at ' + locationList[3] + ' .')
```

```
There are 2830 cars at us-101 .
There are 1484 cars at lankershim .
There are 1531 cars at peachtree .
There are 2955 cars at i-80 .
```

In [25]:
```python
# Directions and movements
print('There are', len(df101.groupby(['Direction'])), 'directions at ' + locationList[0] + ' .')
print('There are', len(df101.groupby(['Movement'])), 'movements at ' + locationList[0] + ' .\n')

print('There are', len(dfLan.groupby(['Direction'])), 'directions at ' + locationList[1] + ' .')
print('There are', len(dfLan.groupby(['Movement'])), 'movements at ' + locationList[1] + ' .\n')

print('There are', len(dfPea.groupby(['Direction'])), 'directions at ' + locationList[2] + ' .')
print('There are', len(dfPea.groupby(['Movement'])), 'movements at ' + locationList[2] + ' .\n')

print('There are', len(df80.groupby(['Direction'])), 'directions at ' + locationList[3] + ' .')
print('There are', len(df80.groupby(['Movement'])), 'movements at ' + locationList[3] + ' .\n')
```

```
There are 0 directions at us-101 .
There are 0 movements at us-101 .

There are 4 directions at lankershim .
There are 3 movements at lankershim .

There are 4 directions at peachtree .
There are 3 movements at peachtree .

There are 0 directions at i-80 .
There are 0 movements at i-80 .
```

Understand the Data - us-101

```python
In [28]: df101.sort_values(by = ['Global_Time']) # We could tell that the number of cars is varied at different times
```

Out[28]:

	Vehicle_ID	Frame_ID	Total_Frames	Global_Time	Local_X	Local_Y	Global_X	Global_Y	v_length	v_Width	...	D_Zone	Int_ID	Section_ID	Direction	Movement	Preceding	Following	Space_Headwa
788	5	8	452	1118846979700	39.788	39.154	6451122.815	1873326.569	17.0	7.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
789	5	9	452	1118846979800	39.767	43.153	6451125.503	1873323.608	17.0	7.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
790	5	10	452	1118846979900	39.747	47.154	6451128.192	1873320.646	17.0	7.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
791	5	11	452	1118846980000	39.726	51.154	6451130.881	1873317.684	17.0	7.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
792	5	12	452	1118846980100	39.705	55.153	6451133.569	1873314.723	17.0	7.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
793	5	13	452	1118846980200	39.685	59.154	6451136.258	1873311.761	17.0	7.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
0	2	13	437	1118846980200	16.467	35.381	6451137.641	1873344.962	14.5	4.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
1	2	14	437	1118846980300	16.447	39.381	6451140.329	1873342.000	14.5	4.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
794	5	14	452	1118846980300	39.665	63.154	6451138.946	1873308.799	17.0	7.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
2	2	15	437	1118846980400	16.426	43.381	6451143.018	1873339.038	14.5	4.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
795	5	15	452	1118846980400	39.643	67.154	6451141.635	1873305.838	17.0	7.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
3	2	16	437	1118846980500	16.405	47.380	6451145.706	1873336.077	14.5	4.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
796	5	16	452	1118846980500	39.623	71.154	6451144.324	1873302.876	17.0	7.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
4	2	17	437	1118846980600	16.385	51.381	6451148.395	1873333.115	14.5	4.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
797	5	17	452	1118846980600	39.603	75.154	6451147.012	1873299.914	17.0	7.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
798	5	18	452	1118846980700	39.581	79.153	6451149.701	1873296.953	17.0	7.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
5	2	18	437	1118846980700	16.364	55.381	6451151.084	1873330.153	14.5	4.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
6	2	19	437	1118846980800	16.344	59.381	6451153.772	1873327.192	14.5	4.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
799	5	19	452	1118846980800	39.562	83.194	6451152.389	1873293.991	17.0	7.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
7	2	20	437	1118846980900	16.323	63.379	6451156.461	1873324.230	14.5	4.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
800	5	20	452	1118846980900	39.541	87.155	6451155.078	1873291.029	17.0	7.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
801	5	21	452	1118846981000	39.520	90.947	6451157.767	1873288.068	17.0	7.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
8	2	21	437	1118846981000	16.303	67.383	6451159.149	1873321.268	14.5	4.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
802	5	22	452	1118846981100	39.544	94.675	6451160.455	1873285.106	17.0	7.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
9	2	22	437	1118846981100	16.282	71.398	6451161.838	1873318.307	14.5	4.9	...	NaN	NaN	NaN	NaN	NaN	0	13	0.0
3333	13	22	432	1118846981100	16.133	35.842	6451138.197	1873344.842	16.0	4.9	...	NaN	NaN	NaN	NaN	NaN	2	0	35.5
803	5	23	452	1118846981200	39.592	98.467	6451163.144	1873282.144	17.0	7.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10	2	23	437	1118846981200	16.262	75.401	6451164.546	1873315.323	14.5	4.9	...	NaN	NaN	NaN	NaN	NaN	0	13	0.0
3334	13	23	432	1118846981200	16.113	39.841	6451140.885	1873341.881	16.0	4.9	...	NaN	NaN	NaN	NaN	NaN	2	0	35.5
804	5	24	452	1118846981300	39.641	102.428	6451165.832	1873279.183	17.0	7.9	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10420875	1942	9926	856	1118849749600	27.170	2148.343	6452679.977	1871948.931	14.5	7.4	...	NaN	NaN	NaN	NaN	NaN	0	1948	0.0
10420876	1942	9927	856	1118849749700	27.166	2153.343	6452683.889	1871945.818	14.5	7.4	...	NaN	NaN	NaN	NaN	NaN	0	1948	0.0
10425462	1948	9927	841	1118849749700	29.855	2048.136	6452599.985	1872009.869	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	1942	0	105.2
10420877	1942	9928	856	1118849749800	27.163	2158.343	6452687.801	1871942.704	14.5	7.4	...	NaN	NaN	NaN	NaN	NaN	0	1948	0.0
10425463	1948	9928	841	1118849749800	30.050	2052.739	6452603.429	1872006.809	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	1942	0	105.6
10420878	1942	9929	856	1118849749900	27.158	2163.343	6452691.714	1871939.591	14.5	7.4	...	NaN	NaN	NaN	NaN	NaN	0	1948	0.0
10425464	1948	9929	841	1118849749900	30.246	2057.341	6452606.872	1872003.749	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	1942	0	106.0
10425465	1948	9930	841	1118849750000	30.442	2061.944	6452610.316	1872000.689	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425466	1948	9931	841	1118849750100	30.637	2066.547	6452613.760	1871997.629	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425467	1948	9932	841	1118849750200	30.833	2071.149	6452617.203	1871994.569	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425468	1948	9933	841	1118849750300	31.028	2075.751	6452620.647	1871991.509	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425469	1948	9934	841	1118849750400	31.224	2080.355	6452624.091	1871988.449	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425470	1948	9935	841	1118849750500	31.419	2084.986	6452627.535	1871985.389	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425471	1948	9936	841	1118849750600	31.622	2089.573	6452630.981	1871982.317	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425472	1948	9937	841	1118849750700	31.825	2094.049	6452634.418	1871979.253	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425473	1948	9938	841	1118849750800	32.029	2098.453	6452637.849	1871976.257	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425474	1948	9939	841	1118849750900	32.148	2102.845	6452641.300	1871973.358	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425475	1948	9940	841	1118849751000	32.180	2107.299	6452644.785	1871970.543	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425476	1948	9941	841	1118849751100	32.168	2111.809	6452648.286	1871967.772	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425477	1948	9942	841	1118849751200	32.148	2116.347	6452651.859	1871964.955	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425478	1948	9943	841	1118849751300	32.147	2120.978	6452655.477	1871962.076	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425479	1948	9944	841	1118849751400	32.165	2125.698	6452659.147	1871959.133	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425480	1948	9945	841	1118849751500	32.165	2130.452	6452662.906	1871956.141	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425481	1948	9946	841	1118849751600	32.182	2135.168	6452666.576	1871953.198	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10422111	1948	9947	841	1118849751700	32.181	2139.799	6452670.194	1871950.320	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10422345	1948	9948	841	1118849751800	32.161	2144.323	6452673.741	1871947.519	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425482	1948	9949	841	1118849751900	32.141	2148.862	6452677.289	1871944.717	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425483	1948	9950	841	1118849752000	32.155	2153.347	6452680.783	1871941.914	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425484	1948	9951	841	1118849752100	32.150	2158.348	6452684.696	1871938.801	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0
10425485	1948	9952	841	1118849752200	32.147	2162.848	6452688.217	1871935.999	14.0	7.4	...	NaN	NaN	NaN	NaN	NaN	0	0	0.0

4679934 rows × 25 columns

```python
In [31]: # What is the max/min number of cars at the same time
maxNum = df101.groupby('Global_Time').size().max()
minNum = df101.groupby('Global_Time').size().min()

print('There are at most', maxNum, 'cars running on the road us-101 at the same time.')
print('There are at least', minNum, 'cars running on the road us-101 at the same time.')
```

```
There are at most 386 cars running on the road us-101 at the same time.
There are at least 1 cars running on the road us-101 at the same time.
```

In [33]:
```python
# How long does these cars run together
timeAndCarNumber = df101.groupby('Global_Time').size()
timeListOfMaxNum = timeAndCarNumber[timeAndCarNumber == 386].index

print(maxNum, 'cars run at the same time for', len(timeListOfMaxNum)/10, 'seconds.' )
```

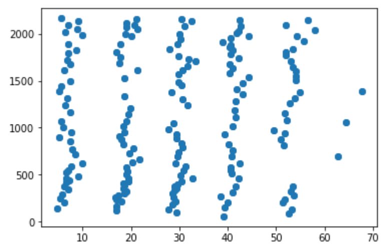
```
386 cars run at the same time for 0.5 seconds.
```

In [34]:
```python
# Plot the situation when 386 cars run together

# Select the first frame
time0 = timeListOfMaxNum[0]

# Get the x, y coordinates of these cars
dfTime0 = df101[df101['Global_Time'] == time0]
carLocList = [[x, y] for x, y in zip(dfTime0.loc[:, 'Local_X'], dfTime0.loc[:, 'Local_Y'])]

# Plot the situation
plt.scatter(*zip(*carLocList))
plt.show()
```



In [35]:
```python
# Define a fucntion to calculate the distance between two cars
def calDistance(carLoc1, carLoc2):
    x1 = carLoc1[0]
    y1 = carLoc1[1]
    x2 = carLoc2[0]
    y2 = carLoc2[1]
    return np.sqrt((x1 - x2)**2 + (y1 - y2)**2)
```

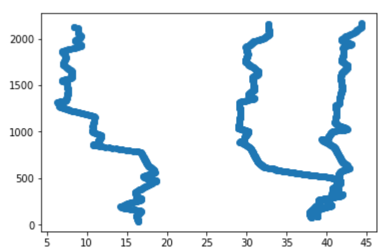
In [39]:
```python
# Find the largest distance between two cars at this moment
maxDistance = 0
for i in range(maxNum - 1):
    for j in range(i + 1, maxNum):
        dis = calDistance(carLocList[i], carLocList[j])
        if dis > maxDistance:
            maxDistance = dis

print('The largest distance between two cars at this moment is %.2f feets.' %maxDistance)
```

```
The largest distance between two cars at this moment is 2109.31 feets.
```

In [40]:
```python
# Visualize the car's track (ID = 2)
dfID2 = df101[df101['Vehicle_ID'] == 2]
CarTrack2 = [[x, y] for x, y in zip(dfID2.loc[:, 'Local_X'], dfID2.loc[:, 'Local_Y'])]

plt.scatter(*zip(*CarTrack2))
plt.show()
```



In [ ]: