Where to Park?

Using Python to Map Parking Regulations Signs to the Curb in Near Real-Time

Jada Macharie, Graduate Student
Department of Geography, Hunter College, CUNY

2024 Annual Meeting of the
Transportation Research Board
9 January 2024

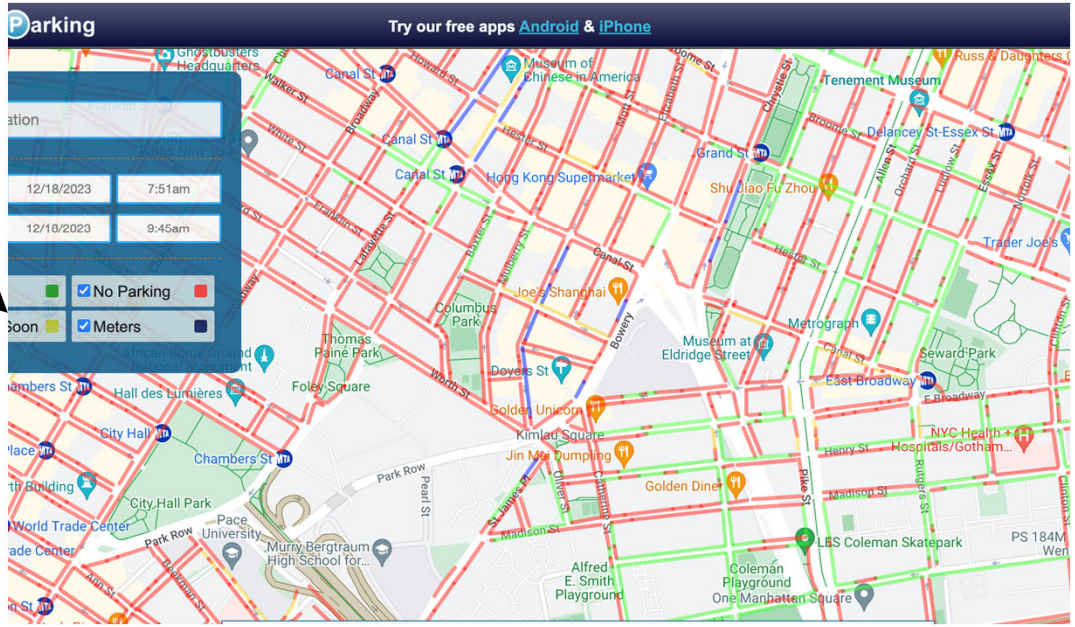
Background

The New York City Department of Transportation is responsible for installing most of the signage in the five boroughs of NYC.

https://data. cityofnewyork.us/Transportation/Parking-Regulation-Locations-and-Signs/xswq-wnv9

Goals & Objectives

The data used to create DOT's interactive map, will be used to a dynamic map that is like smooth parking's web map.



Scope → Datasets → Software

CB1, Financial District

NYC Open Data

Pavement Edge
Pedestrian Ramps
Parking Regulations

NYC OpenData

python™

QGIS

PostgreSQL

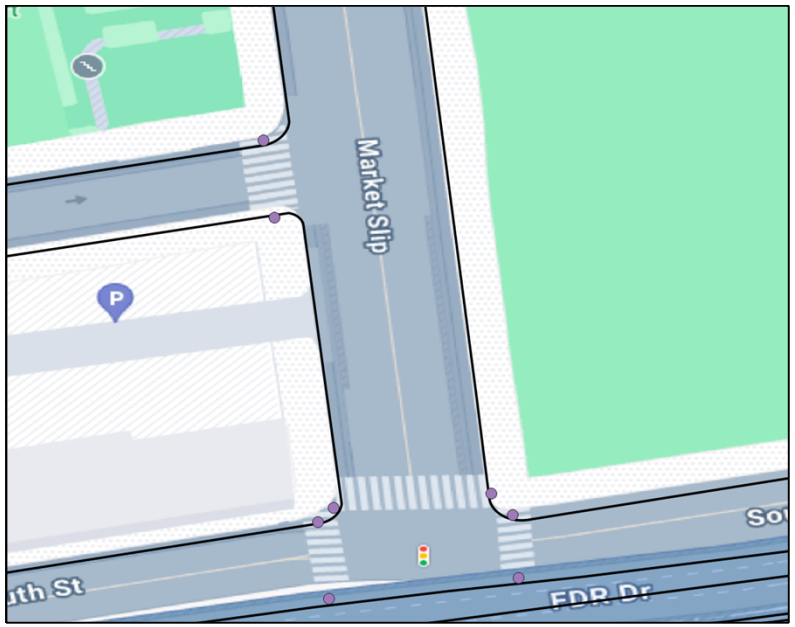
QGIS COMPONENT

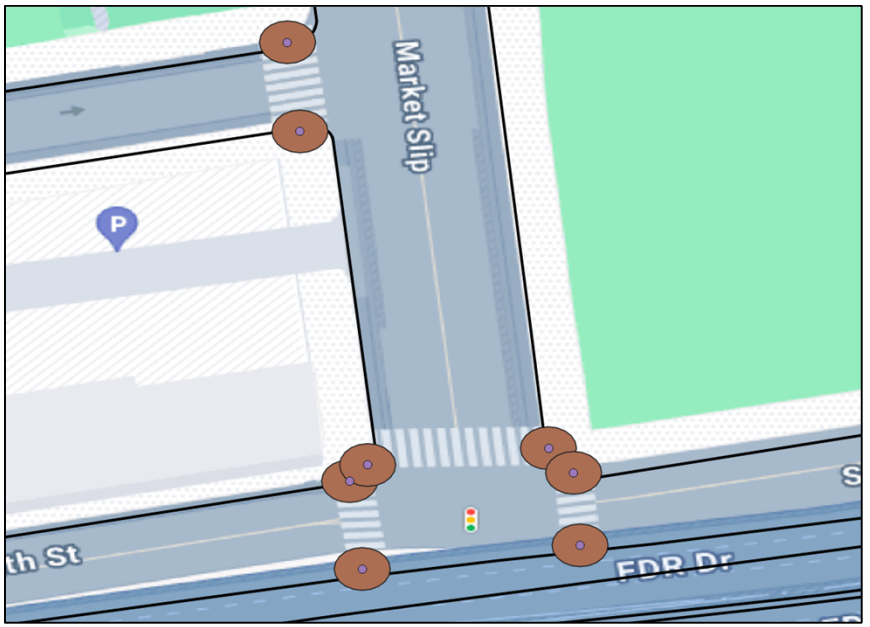
Methods: Snap Points

Methods: Snap Points

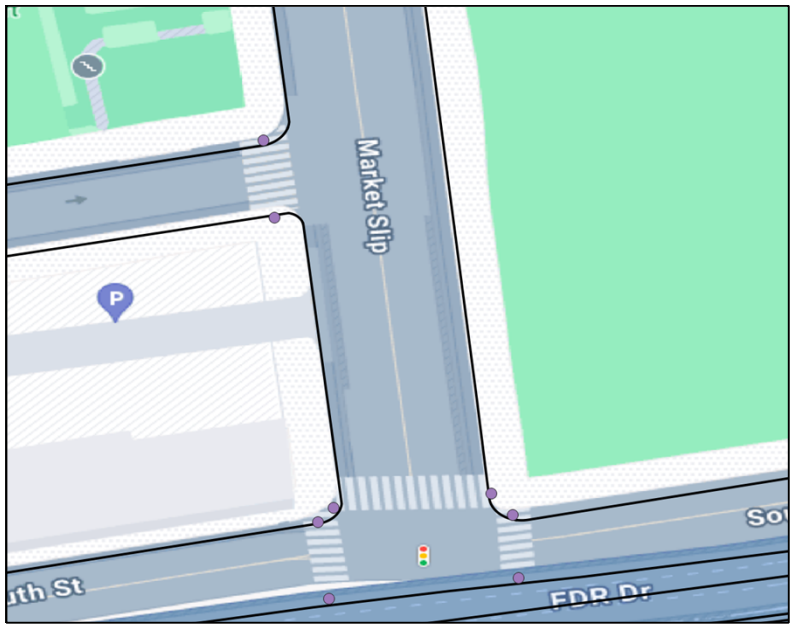

```
156  UPDATE "DOT"."ParkingSigns" AS points
157  SET geom = ST_ClosestPoint("DOT".ClippedPave".geom, points.geom)
158  FROM "DOT"."ClippedPave" AS line
159  WHERE ST_DWithin(line.geom, points.geom, 0.001);
```

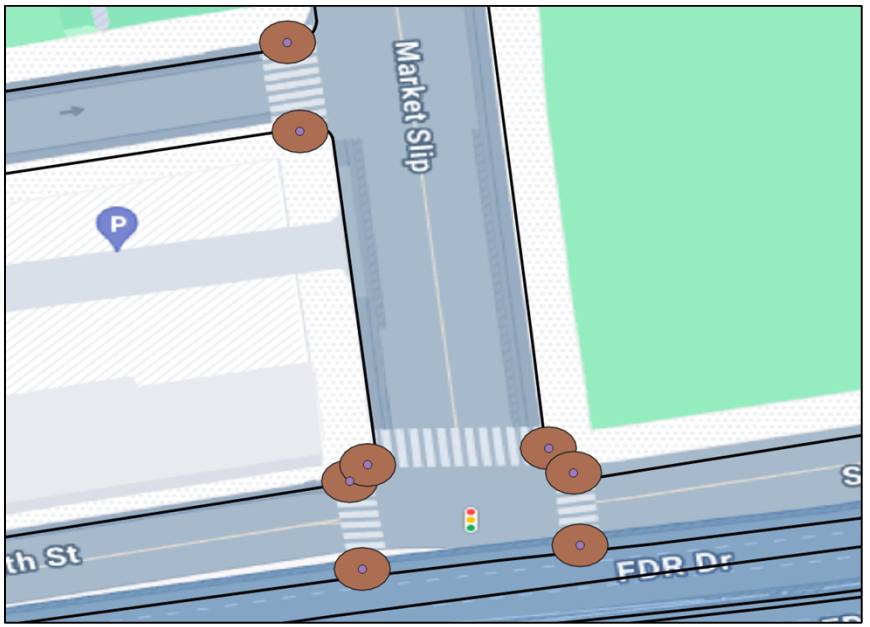
Methods: Buffering





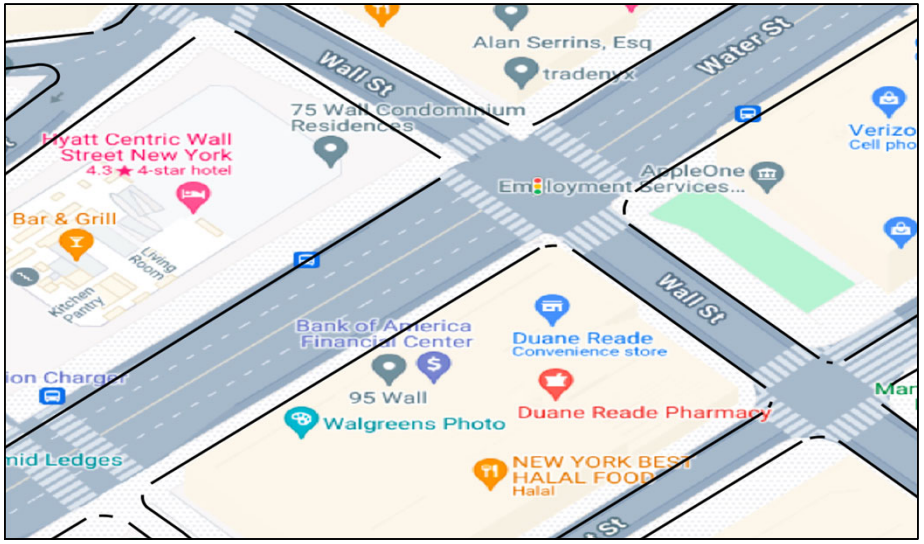
Methods: Buffering



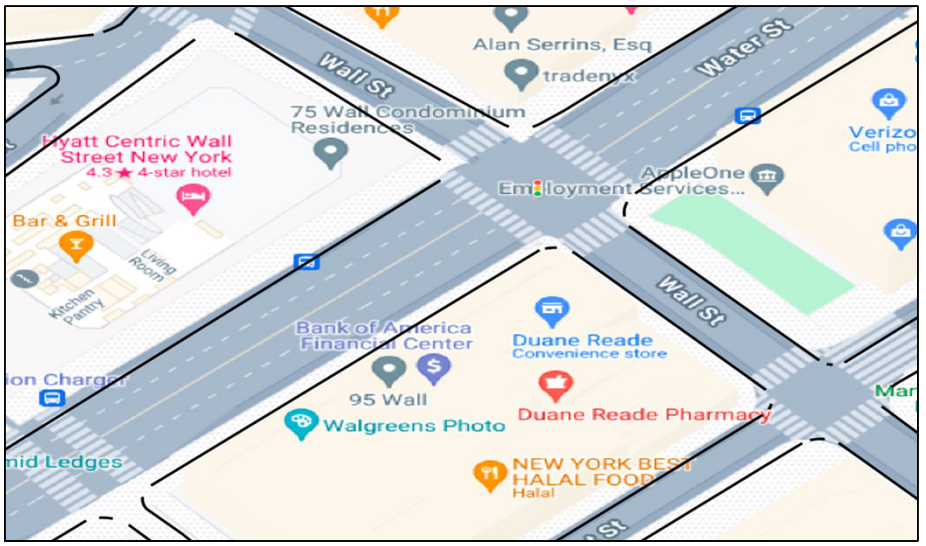



```
169  -- Create a new table to store the buffered points
170  CREATE TABLE buffered_points AS
171  SELECT
172      id, -- Replace with your identifier column
173      ST_Buffer(newgeom, 10) AS buffered_geom
174  FROM
175      "DOT"."PedRamps";
176
```

Methods: Clip



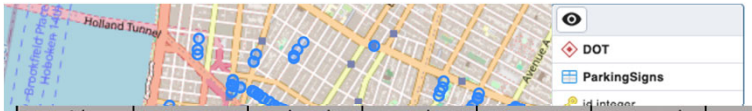
Methods: Clip




```sql
-- Create a new table to store the clipped line segments
CREATE TABLE clipped_segments AS
SELECT
    cp.id AS segment_id,
    bp.id AS point_id,
    ST_Intersection(cp.newgeom, bp.buffered_geom) AS clipped_geom
FROM
    "DOT"."ClippedPave" cp
JOIN
    buffered_points bp ON ST_Intersects(cp.geom, bp.buffered_geom);

```

SQL COMPONENT

Variables and Assumptions

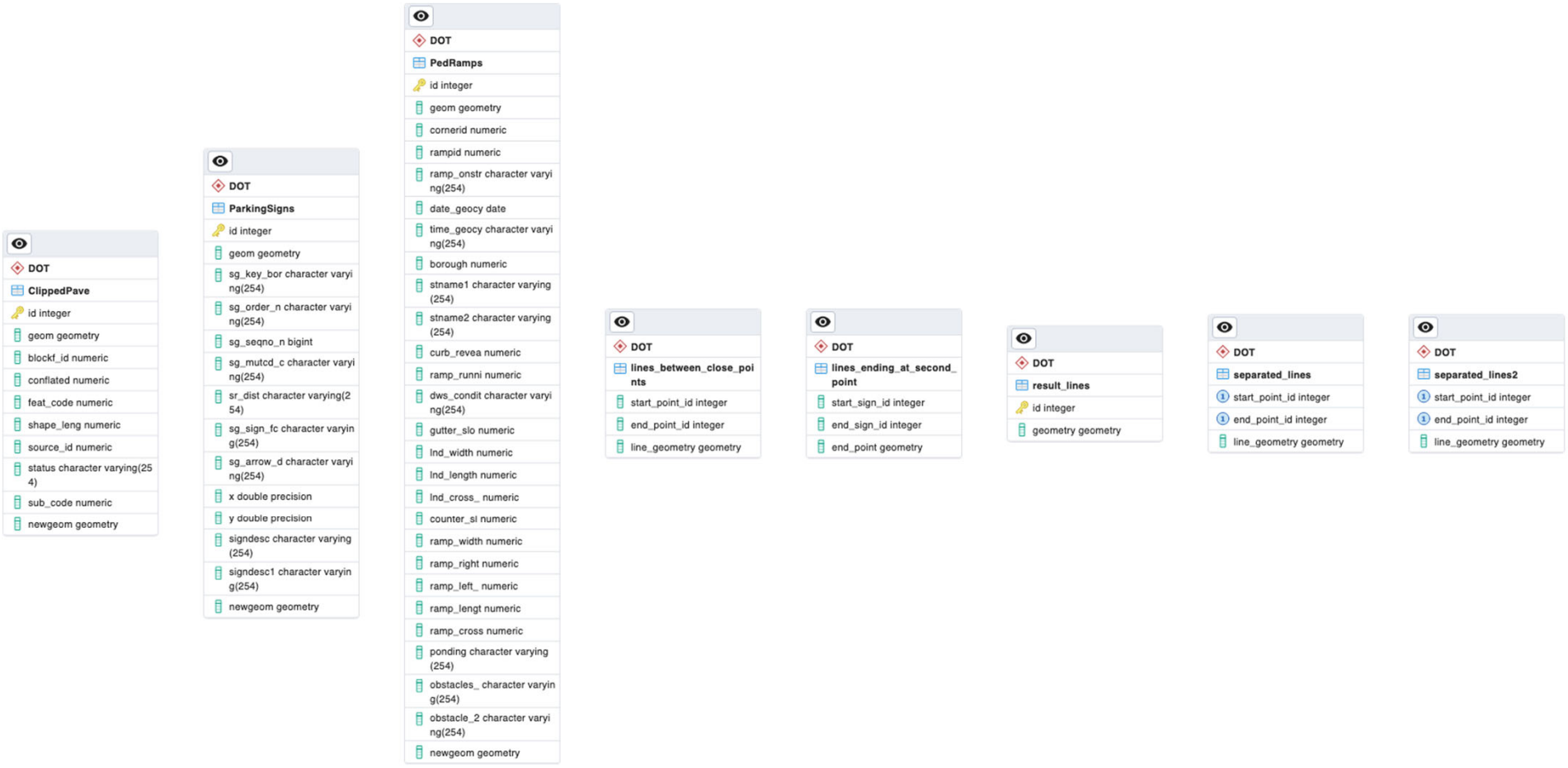


id	geom	sg_key_bor	sg_order_n	sg_seqno_n	sg_mutcd_c	sr_dist	sg_sign_fc	sg_arrow_d	x	y	signdesc	signdesc1	newgeom
411	0101000020	M	P-510220	9	PS-9A	NULL	NULL	NULL	983165	202507	PAY-BY-CELL LOCATOR NUMBER	PAY-BY-CELL LOCATOR NUMBER	0101000020E6
1	0101000020	M	P-00201628	3	PS-83B	NULL	NULL	NULL	986732	200588	NO PARKING (SANITATION BROOM SY	NO PARKING (SANITATION BROOM SY	0101000020E6
2	0101000020	M	P-00201628	5	PS-9A	NULL	NULL	NULL	986732	200588	PAY-BY-CELL LOCATOR NUMBER	PAY-BY-CELL LOCATOR NUMBER	0101000020E6
3	0101000020	M	P-00201628	6	PS-83B	NULL	NULL	NULL	986796	200563	NO PARKING (SANITATION BROOM SY	NO PARKING (SANITATION BROOM SY	0101000020E6
4	0101000020	M	P-00201628	8	PS-9A	NULL	NULL	NULL	986796	200563	PAY-BY-CELL LOCATOR NUMBER	PAY-BY-CELL LOCATOR NUMBER	0101000020E6
5	0101000020	M	P-00622233	4	PS-9A	NULL	NULL	NULL	984197	201202	PAY-BY-CELL LOCATOR NUMBER	PAY-BY-CELL LOCATOR NUMBER	0101000020E6
6	0101000020	M	P-00622233	5	PS-269CA	NULL	NULL	N	984154	201138	3 HMP COMMERCIAL VEHICLES ONLY 7	3 HMP COMMERCIAL VEHICLES ONLY 7	0101000020E6
7	0101000020	M	P-00622233	6	PS-9A	NULL	NULL	NULL	984154	201138	PAY-BY-CELL LOCATOR NUMBER	PAY-BY-CELL LOCATOR NUMBER	0101000020E6
8	0101000020	M	P-00622233	9	PS-1GA	NULL	NULL	N	984092	201072	NO STANDING ANYTIME --> (SUPERSE	NO STANDING ANYTIME -->	0101000020E6
412	0101000020	M	P-510229	5	PS-9A	NULL	NULL	NULL	982959	202639	PAY-BY-CELL LOCATOR NUMBER	PAY-BY-CELL LOCATOR NUMBER	0101000020E6
9	0101000020	M	P-00201628	4	PS-54C	NULL	NULL	NULL	986732	200588	2 HOUR METERED PARKING 9AM-7PM	2 HOUR METERED PARKING 9AM-7PM	0101000020E6
10	0101000020	M	P-00201628	7	PS-54C	NULL	NULL	NULL	986796	200563	2 HOUR METERED PARKING 9AM-7PM	2 HOUR METERED PARKING 9AM-7PM	0101000020E6
11	0101000020	M	P-00622233	3	PS-269C	NULL	NULL	NULL	984197	201202	3 HMP COMMERCIAL VEHICLES ONLY 7	3 HMP COMMERCIAL VEHICLES ONLY 7	0101000020E6
12	0101000020	M	P-00622233	7	PS-11GA	NULL	NULL	N	984135	201111	NO STANDING HOTEL LOADING ZONE	NO STANDING HOTEL LOADING ZONE	0101000020E6
13	0101000020	M	P-01253794	7	PS-174C	NULL	NULL	NULL	982219	196251	2 HMP 8:30AM-10PM EXCEPT SUNDAY	2 HMP 8:30AM-10PM EXCEPT SUNDAY	0101000020E6
14	0101000020	M	P-01253794	3	PS-167BA	NULL	NULL	N	982309	196333	NO PARKING (SANITATION BROOM SY	NO PARKING (SANITATION BROOM SY	0101000020E6
15	0101000020	M	P-01253794	4	PS-174CA	NULL	NULL	N	982309	196333	2 HMP 8:30AM-10PM EXCEPT SUNDAY	2 HMP 8:30AM-10PM EXCEPT SUNDAY	0101000020E6
16	0101000020	M	P-01253794	6	PS-167B	NULL	NULL	NULL	982219	196251	NO PARKING (SANITATION BROOM SY	NO PARKING (SANITATION BROOM SY	0101000020E6

Pavement Edge

Pedestrian Ramps

Database Analysis



Database Analysis

schemaname	tablename	attname	inherited	null_frac	avg_width	n_distinct	most_common_vals	most_common_freqs	histogram_bounds	correlation
DOT	PedRamps	id	FALSE	0	4	-1			{1,45,90,135,180,225,270	0.999995
DOT	PedRamps	geom	FALSE	0	32	-1			{0101000020E610000B	0.18551752
DOT	PedRamps	cornerid	FALSE	0	7	-0.635452	{1086837,1174286,100	{0.00089186174,0.00089	{1000001,1083999,1084	0.044356387
DOT	PedRamps	rampid	FALSE	0	7	-1			{13155,13353,23676,239	0.9631088
DOT	PedRamps	ramp_onstr	FALSE	0.000223	16	-0.133333	{"EAST HOUSTON STREET	{0.02787068,0.02474916	{"1 PLACE","3 PLACE","7 A	0.009574794
DOT	PedRamps	date_geocy	FALSE	0	4	39	{2018-10-07,2018-10-0	{0.13913043,0.10078038	{2017-06-26,2018-09-15	0.39134756
DOT	PedRamps	time_geocy	FALSE	0	13	1	{00:00:00.000}	{1}		1
DOT	PedRamps	borough	FALSE	0	5	1	{1}	{1}		1
DOT	PedRamps	stname1	FALSE	0	14	250	{"EAST HOUSTON STREET	{0.032329988,0.0254180	{"1 PLACE","3 PLACE","7 A	0.009657429
DOT	PedRamps	stname2	FALSE	0.007135	14	242	{"GRAND STREET",BROAD	{0.02497213,0.02408026	{"1 PLACE","2 PLACE","3 P	0.02173929
DOT	PedRamps	curb_revea	FALSE	0	5	38	{0.6,0.5,0.7,0.4,0.3,0.8,0	{0.1270903,0.109253064	{0.12,3,4}	0.059821106
DOT	PedRamps	ramp_runni	FALSE	0	7	235	{999,7.0000000000000	{0.055072464,0.0153846	{-7.3,-3.4,-2.2,-1.7,-1.3,-1,	0.061340097
DOT	PedRamps	dws_condit	FALSE	0	12	6	{"Good Condition",Miss	{0.6749164,0.29453734,0	0.026978819,0.002229654	0.46806994
DOT	PedRamps	gutter_slo	FALSE	0	6	149	{999,0.3,0.4,0.6,-0.3,0.2	{0.054849498,0.0238573	{-11.9,-9.5,-8.5,-7.5,-7.000	0.08175364
DOT	PedRamps	lnd_width	FALSE	0	6	-0.125753	{999,48,48.6,48.2,47.7,4	{0.05529543,0.02497213	{11.1,25.8,28.9,30.6,32.5,	0.035623975
DOT	PedRamps	lnd_length	FALSE	0	5	256	{60,999,888,48,555,54.2	{0.8541806,0.05529543,(	{0.2,0.3,1.8,2.900000000	0.7752019
DOT	PedRamps	lnd_cross_	FALSE	0	6	174	{999,0.4,0.3,-0.4,1.1,-0.2	{0.055072464,0.0227424	{-23,-12.2,-9.9,-8.5,-8.3,-7	0.07554411
DOT	PedRamps	counter_sl	FALSE	0	7	264	{999,-3.6999999999999	{0.054849498,0.0113712	{-27.6,-13.3,-12.8,-12.2,-1	0.11576291
DOT	PedRamps	ramp_width	FALSE	0	6	-0.168785	{48,47.6,47.9,47.1,48.1,	{0.018729096,0.0127090	{22.3,29.4,32.5,33.59999	0.058442097
DOT	PedRamps	ramp_right	FALSE	0	7	434	{999,9.9,8.7,8.9,8.8,9.1,9	{0.055518396,0.0120401	{-51.6,-4,-2.300000000000	0.04416851
DOT	PedRamps	ramp_left_	FALSE	0	6	-0.151394	{60,56.7,61.5,57.8,60.3,	{0.008918618,0.0080267	{18,28.1,31.9,33.4,34.3,35	-0.008241484
DOT	PedRamps	ramp_lengt	FALSE	0	7	-0.112821	{999,-9.2000000000000	{0.05574136,0.00958751	{-95.5,-43.1,-39.3,-37,-35	0.086833104
DOT	PedRamps	ramp_cross	FALSE	0	6	166	{999,-0.3,0.1,0.2,0.3,-0.1	{0.055072464,0.0238573	{-17.6,-14.7,-11,-9.7,-9,-8	0.066083856
DOT	PedRamps	ponding	FALSE	0	3	2	{No,Yes}	{0.78283167,0.21716835}		0.5895427
DOT	PedRamps	obstacles_	FALSE	0	5	12	{None,"Utility access",Bo	{0.94938684,0.03723523,	0.008472687,0.00089186	0.8840026
DOT	PedRamps	obstacle_2	FALSE	0	6	20	{None,"Utility access","U	{0.85462654,0.11817168	{"Fence,Trees,Landscape",	0.7535288
DOT	lines_between_close_points	start_point_id	FALSE	0	4	278	{2,9,3,5,15,14,4,18,24,1	{0.0074,0.0072333333,0.	{95,95,100,101,102,105,1	0.998807
DOT	lines_between_close_points	end_point_id	FALSE	0	4	278	{276,279,277,272,278,2	{0.0075,0.0075,0.007433	{2,19,26,32,37,41,45,48,5	0.4744194
DOT	lines_between_close_points	line_geometry	FALSE	0	48	-1	{0102000020E6100000	{0.0028,0.0026666666,0.	{0102000020E61000000	-0.07602879
DOT	lines_ending_at_second_point	start_sign_id	FALSE	0	4	277	{2,9,24,12,1,8,10,4,14,3,	{0.008066666,0.0078666	{74,74,81,88,91,92,95,98	0.99854845
DOT	lines_ending_at_second_point	end_sign_id	FALSE	0	4	276	{271,276,264,278,270,2	{0.007866667,0.0078666	{4,18,26,31,36,41,45,49,5	0.4701274
DOT	lines_ending_at_second_point	end_point	FALSE	0	32	-1	{0101000020E6100000	{0.06846666,0.05973333	{0101000020E61000001	0.12126758
DOT	result_lines	id	FALSE	0	4	-1			{51,55378,117657,18206	1
DOT	result_lines	geometry	FALSE	0	48	-1	{0102000000020000000	{0.0020333333,0.0018,0.	{0102000000020000000D	0.008366105
DOT	separated_lines2	start_point_id	FALSE	0	4	-1			{1,125,251,371,517,632,7	0.9999331
DOT	separated_lines2	end_point_id	FALSE	0	4	-0.845588	{8095,9920,9814,4537,(	{0.0021626297,0.001946	{3,134,316,474,602,729,8	0.88289005
DOT	separated_lines2	line_geometry	FALSE	0	48	-1	{0102000020E6100000	{0.0047577852,0.003676	{0102000020E61000000	-0.15370035
DOT	ParkingSigns	id	FALSE	0	4	-1			{1,100,199,299,398,498,5	0.9996314
DOT	ParkingSigns	geom	FALSE	0	32	-1	{0101000020D7080000	{0.0023131853,0.002112	{0101000020D70800009	0.000242363
DOT	ParkingSigns	sg_key_bor	FALSE	0	2	1	{M}	{1}		1
DOT	ParkingSigns	sg_order_n	FALSE	0	10	-0.207282	{P-01346902,S-343171,	{0.00553153,0.00462637	{P-00201628,P-0128645	0.9197458
DOT	ParkingSigns	sg_seqno_n	FALSE	0	8	58	{3,4,5,6,7,8,9,10,11,2,12	{0.19269837,0.1802273,(	{46,50,51,52,53,54,55,56,	-0.13906959
DOT	ParkingSigns	sg_mutcd_c	FALSE	0	7	802	{PS-1G,PS-9A,PS-1GA,PS	{0.083978675,0.0751282	{INSERT,PS-101C,PS-105D	-0.07989143
DOT	ParkingSigns	sr_dist	FALSE	0	5	1	{"NULL"}	{1}		1
DOT	ParkingSigns	sg_sign_fc	FALSE	0	4	6	{"NULL",W,E,N,NS,S}	{0.9950719,0.0017097455	0.001106306,0.0010057	0.98623794
DOT	ParkingSigns	sg_arrow_d	FALSE	0	4	7	{"NULL",S,N,W,E,EW,NS}	{0.75379664,0.06708237,	0.06215428,0.05883536,	0.5799674
DOT	ParkingSigns	x	FALSE	0	8	-0.518254	{983091,984333,98010	{0.0023131853,0.002112	{979286.506989,979576	-0.11256033
DOT	ParkingSigns	y	FALSE	0	8	-0.504073	{198563,198597,19959	{0.0023131853,0.002313	{194834,195319.595883	-0.027601996
DOT	ParkingSigns	signdesc	FALSE	0	70	796	{"NO STANDING ANYTIM	{0.083978675,0.0751282	{"(278) BKLYN-BATTERY TU	0.25076747
DOT	ParkingSigns	signdesc1	FALSE	0	46	784	{"NO STANDING ANYTIM	{0.083978675,0.0751282	{"(278) BKLYN-BATTERY TU	0.24935912
DOT	ParkingSigns	newgeom	FALSE	0	32	-1	{0101000020E6100000	{0.0023131853,0.002112	{0101000020E61000000	-0.17162336

Methods: Connections

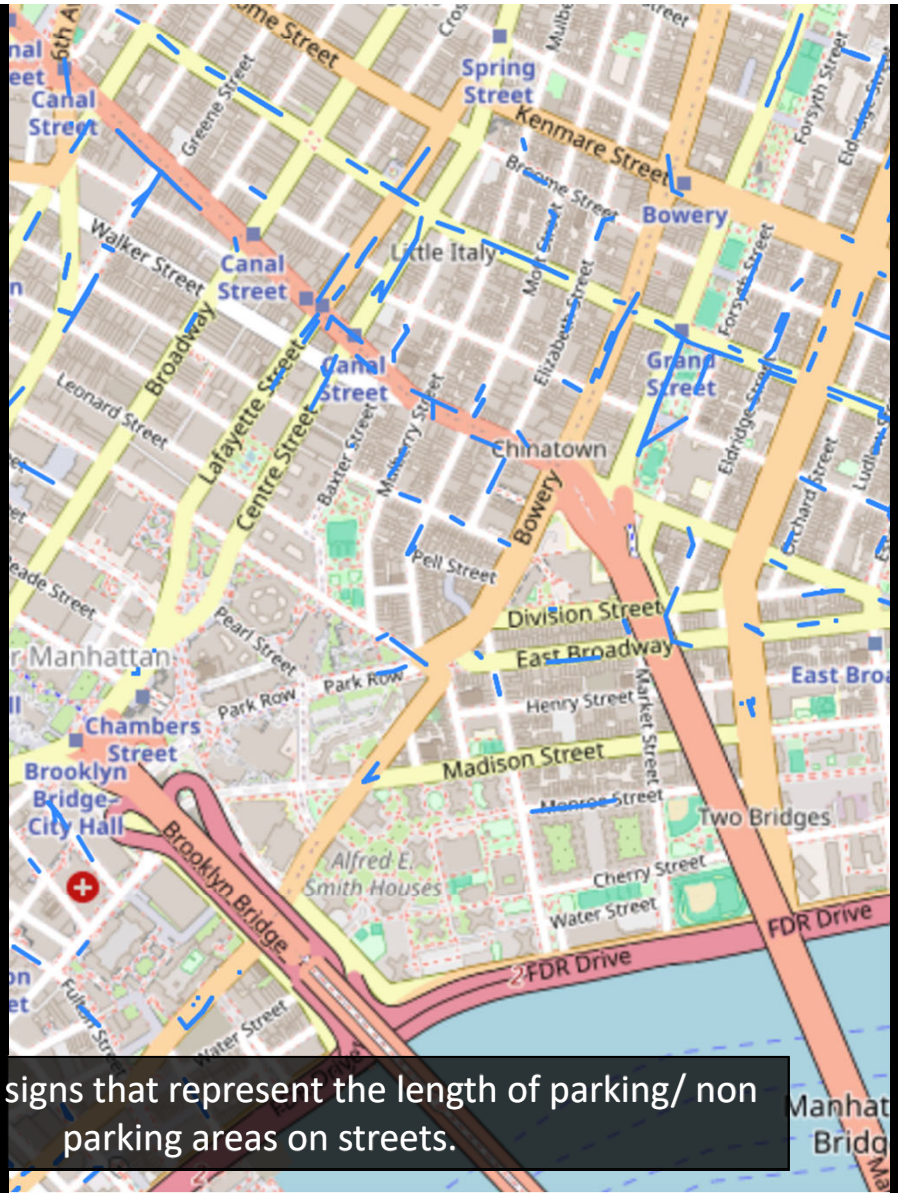
- Reprojected Geometry Column from ESPG:2263 to 4326
- Created Spatial Index to shorter query run time
- Joined Pavement Edge and Parking Regulations using new geometry column
- Ran a query that took the common direction of one point on the same line segment and connected the points in that same direction

```
1  ----------------------------FINAL PROJECT----------------------
2
3  SELECT * FROM "ClippedPave"
4  SELECT * FROM "CroppedPaveEdge"
5
6  ALTER TABLE "ClippedPave"
7  ADD COLUMN newgeom Geometry(MultiLineString, 4326);
8
9  UPDATE "ClippedPave"
10 SET newgeom = ST_Transform(geom, 4326);
11
12 SELECT * FROM "ParkingSigns"
13
14 ALTER TABLE "ParkingSigns"
15 ADD COLUMN newgeom Geometry(Point, 4326);
16
17 UPDATE "ParkingSigns"
18 SET newgeom = ST_Transform(geom, 4326);
19
20
```

```
21 --Spatial Index
22 CREATE INDEX parkingsigns_spatial_index
23 ON "ParkingSigns"
24 USING GIST(newgeom);
25
26 CREATE INDEX paveedge_spatial_index
27 ON "ClippedPave"
28 USING GIST(newgeom);
```

```
107 CREATE TABLE separated_lines2 AS (
108     SELECT
109         p1.id AS start_point_id,
110         p2.id AS end_point_id,
111         ST_MakeLine(p1.newgeom, ST_Transform(p2.closest_geom, 4326)) AS line_geometry
112     FROM
113         "ParkingSigns" p1
114     JOIN LATERAL (
115         SELECT
116             p2.id,
117             ST_Distance(p1.newgeom, p2.newgeom) AS distance,
118             p2.newgeom AS closest_geom,
119             MAX(p2.SG_ARROW_D) AS common_direction  -- Use MAX to get the common direction
120         FROM
121             "ParkingSigns" p2
122         WHERE
123             p1.id != p2.id
124             AND p1.signdesc = p2.signdesc  -- Add condition for matching description
125         GROUP BY
126             p2.id, p2.newgeom
127         ORDER BY
128             ST_Distance(p1.newgeom, p2.newgeom)
129         LIMIT 1
130     ) p2 ON true
131     WHERE
132         ST_DWithin(p1.newgeom, ST_Transform(p2.closest_geom, 4326), 0.001)
133         AND NOT EXISTS (
134             SELECT 1
135             FROM "ClippedPave" s
136             WHERE ST_Intersects(ST_MakeLine(p1.newgeom, ST_Transform(p2.closest_geom, 4326)), ST_Transform(s.geom, 4326))
137         )
138 );
```

Results



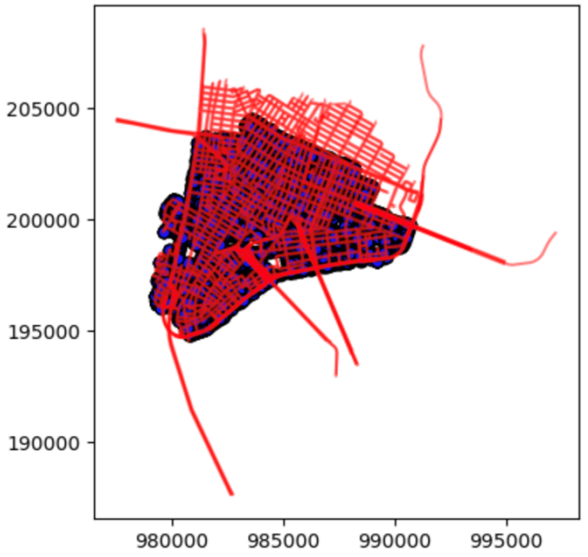
Lines between signs that represent the length of parking/ non parking areas on streets.

PYTHON COMPONENT

Datasets

Pavement Edge

Parking Signs



Parking Signs overlayed on Pavement Edge

CREATE LINES BETWEEN SIGNS USING PROXIMITY, TYPE, & DIRECTION

STEP 1: IMPORT NECESSCARY PACKAGES

[13]:
```
import pandas as pd
import geopandas as gpd
import gmplot
import matplotlib.pyplot as plt
import plotly.express as px

from shapely.geometry import MultiLineString, LineString, Point
from math import radians, sin, cos, sqrt, atan2
```

STEP 2: IMPORT DATA

[7]:
```
# IMPORT SHAPEFILES
Signs = gpd.read_file('/Users/jadamacharie/Downloads/DOT Internship/CommunityBoard1/SnappedPoints.shp')
print(Signs.columns.tolist())
print(Signs.crs)
# Signs.head()
Signs
```

[10]:
```
Edge = gpd.read_file('/Users/jadamacharie/Downloads/DOT Internship/CommunityBoard1/UseCroppedPaveEdge.shp')
Edge.head()
```

[10]:

	blockf_id	conflated	feat_code	shape_leng	source_id	status	sub_code	geometry
0	122260309.0	1.0	2260.0	296.393696	1.222600e+10	Unchanged	226000.0	LINESTRING (989639.410 198318.832, 989626.845 ...
1	122260709.0	1.0	2260.0	434.117494	1.222601e+10	Unchanged	226000.0	LINESTRING (983445.853 200097.374, 983440.315 ...
2	122260055.0	1.0	2260.0	31.256344	1.222600e+10	Updated	226000.0	LINESTRING (982061.120 202854.831, 982060.617 ...
3	122260518.0	1.0	2260.0	170.491174	1.222601e+10	Unchanged	226000.0	LINESTRING (983230.736 200965.158, 983233.711 ...
4	122260543.0	0.0	2260.0	10.151556	1.222601e+10	Unchanged	226000.0	LINESTRING (985882.476 201679.109, 985883.156 ...

STEP 3: SPATIALY CLEAN DATA

[11]:
```
# #CHANGE PROJECTIONS
Signs.crs
{'init': 'epsg:2263'}
print(Signs.crs)

Edge.crs
{'init': 'epsg:2263'}
Edge.crs
```

Method: Initial Steps

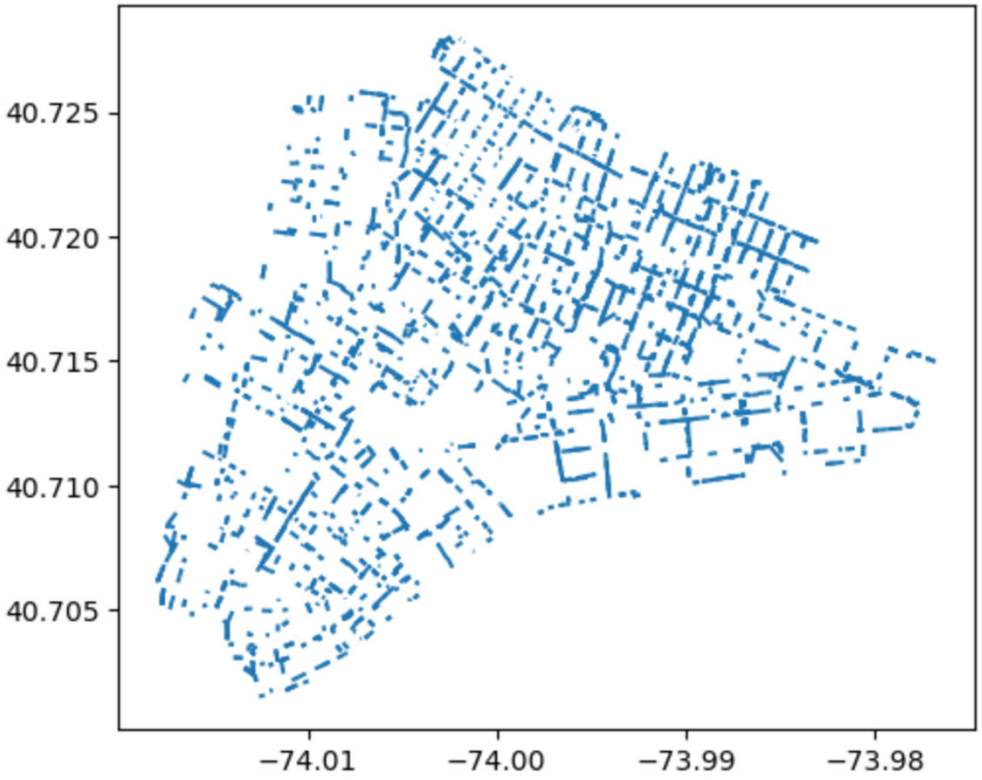
Method: Creating No Parking/ Parking Lengths Between Signs

STEP 5: CREATE LINES BETWEEN SIGN DATA

[55]:

```
from sqlalchemy import create_engine, MetaData, Table, Column, Integer, select, text
from geoalchemy2 import Geometry

# Replace 'your_database_url' with the actual URL of your database
database_url = 'postgresql://jadamacharie:Animalplanet8@localhost:5432/postgres'
engine = create_engine(database_url)

# Define the metadata and the table
metadata = MetaData()

separated_lines2 = Table(
    'separated_lines2', metadata,
    Column('start_point_id', Integer),
    Column('end_point_id', Integer),
    Column('line_geometry', Geometry('LINESTRING')),
)

sql_query = """
INSERT INTO "DOT"."separated_lines2" (start_point_id, end_point_id, line_geometry)
SELECT
    p1.id AS start_point_id,
    p2.id AS end_point_id,
    ST_MakeLine(p1.newgeom, ST_Transform(p2.closest_geom, 4326)) AS line_geometry
FROM
    "DOT"."ParkingSigns" p1
JOIN LATERAL (
    SELECT
        p2.id,
        ST_Distance(p1.newgeom, p2.newgeom) AS distance,
        p2.newgeom AS closest_geom,
        MAX(p2.SG_ARROW_D) AS common_direction
    FROM
        "DOT"."ParkingSigns" p2
    WHERE
        p1.id != p2.id
        AND p1.signdesc = p2.signdesc
    GROUP BY
        p2.id, p2.newgeom
    ORDER BY
        ST_Distance(p1.newgeom, p2.newgeom)
    LIMIT 1
) p2 ON true
WHERE
    ST_DWithin(p1.newgeom, ST_Transform(p2.closest_geom, 4326), 0.001)
    AND NOT EXISTS (
        SELECT 1
        FROM "DOT"."ClippedPave" s
        WHERE ST_Intersects(ST_MakeLine(p1.newgeom, ST_Transform(p2.closest_geom, 4326)), ST_Transform(s.geom, 4326))
    )
    AND NOT EXISTS (
        SELECT 1
        FROM "DOT"."separated_lines2" sl2
        WHERE sl2.start_point_id = p1.id
        AND sl2.end_point_id = p2.id
    )
"""

# Execute the query
with engine.connect() as connection:
    result = connection.execute(text(sql_query))

# Commit the changes
metadata.create_all(engine)
```

Results



Discussion

- Limitations:
  - Transforming my Parking Signs to line data types
  - Negating fire hydrants, Bus stops, loading docks, etc.
  - Some of the Pedestrian Ramps are not located on the corners of the streets
- What did not work:
  - Creating lines between Parking Signs using Pavement Edge
  - Converting all Parking Signs to lines
  - Clipping Pavement Edge by Sign Points

- Next Steps:
  - Color Code line connections based on regulation types/descriptions

jada.macharie96@myhunter.cuny.edu

QUESTIONS?