Automated Extraction of Parking Signs

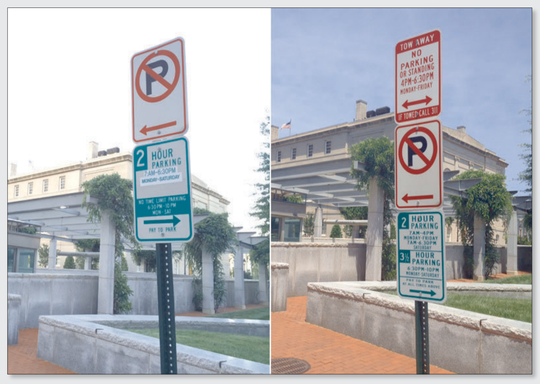
James K. Graham
Osheen Safarian

District Department of Transportation
Washington, DC

Background

Understanding where and when it is safe to park your car is not always easy.
In Washington, DC the variety and complexity of parking sign information is well known and this has lead to two major complaints:

Totem Pole Effect
Parking signs are too complex, causing confusion.



Conflicting Signs
Parking information on one sign conflicts with other parking signs down the block

What is needed is a way to easily visualize parking information at the location in question. This will not only provide better clarity for parking availability to the general public but will also help DDOT to visualize errors in the current asset inventory (conflicting signs).

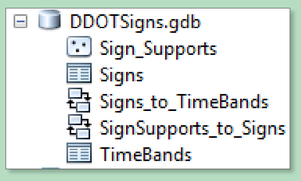
This project seeks to demonstrate how GIS can be used to make parking information more intelligible by displaying it spatially and temporally. To accomplish this, we had to transform traditional GIS data into a new form which makes visualization, query and analysis much more straightforward.

Data

In order to derive parking zones, it's best to go to the source (if you have it): a GIS-based parking sign inventory.

In 2016, DDOT captured over 203,000 street signs within the District of Columbia. Over 30% of the signage was related to parking (permissive or restrictive). That's a lot of parking signs!

Below is the basic structure of our sign inventory data.



Supports

Sign supports indicate where the pole is located and are the only spatial component of our sign inventory.

Signs

This table contains all the information about each sign, including MUTCD/sign type, arrow direction and sign order.

Signs are a related 'child' table to the supports point feature class above (one support - many signs).

Time Bands (Parking Restrictions)

This table contains the day and time restrictions posted on each sign.

Time Bands are a related 'child' table to the signs table above (one sign - many time bands).

Results

Once the parking zones are generated as linear referenced lines, the data are remarkably fun to work with.

Our plan is to continue to work on different ways to help residents get clear and accurate information about parking in Washington, DC. Below is an example of what we have accomplished so far.

Item A depicts summary parking information based upon the day of the week, for each side of the roadway.

Item B depicts what we would like to do next. We would like to transform our parking zone data into a graphic like this.

B

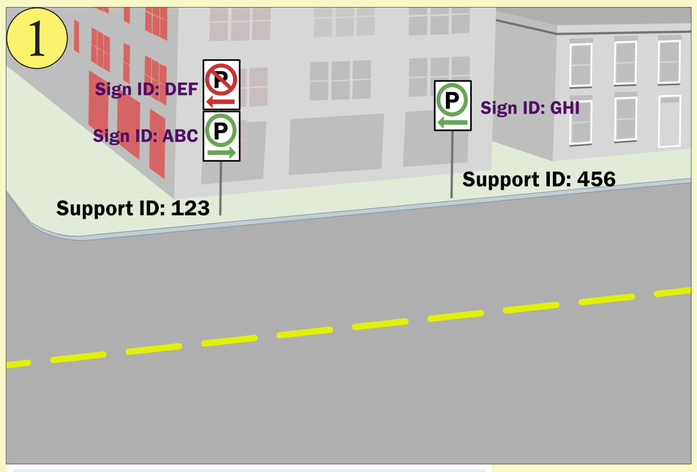
Parking Zone Extraction

This panel shows the approach we use to make parking zones from our sign data.

Sign Inventory

For this example, we have two supports and a total of three signs. For the parking zone build, we focus on the sign table, which contains both unique IDs for each of the signs and the supports they are associated to.

Below the graphic is a table representing the sign data in the graphic.

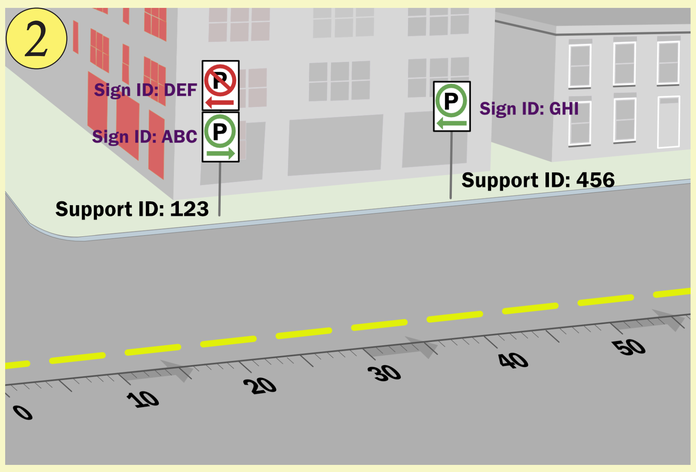


Support ID	Sign ID	SignType	ArrowDirection
123	ABC	R7-22 (Parking)	Right
123	DEF	R8 (No Parking)	Left
456	GHI	R7-22 (Parking)	Left

Linear Referencing System (LRS)

Centerlines play a key role in the creation of parking zones. Our centerlines have linear referencing system M-values (measure values), which ascend in the digitized direction of the polyline.

Below is a graphical example depicting the measure values along First St.

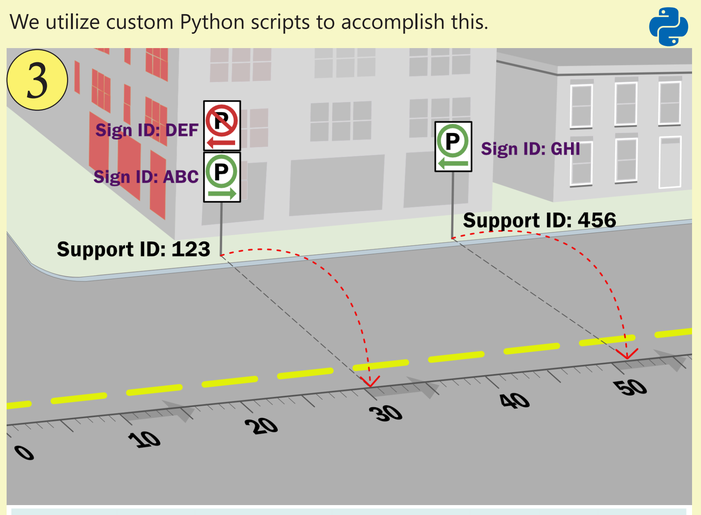


Get Linear Reference from Centerline

For every parking sign in our inventory, we linearly reference it to our centerline routes. This provides 3 key pieces of data which are necessary to build zones: Street/route, Side of Street and LRS Measure.

Below the graphic is an updated view of the sign table showing the change.

We utilize custom Python scripts to accomplish this.

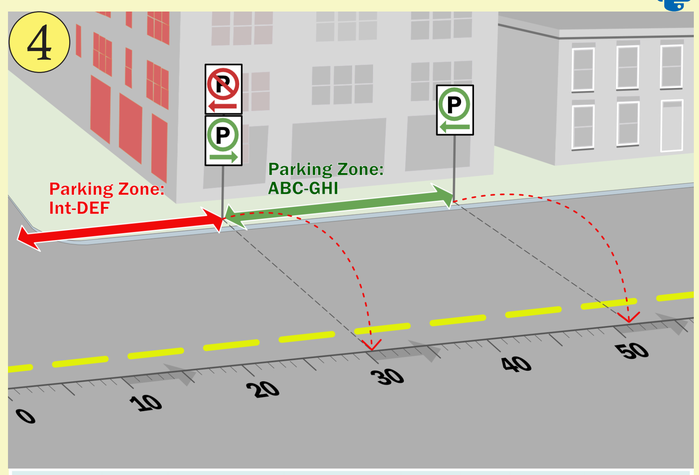


Support ID	Sign ID	SignType	ArrowDirection	Street	Side	Measure
123	ABC	R7-22 (Parking)	Right	1st	Left	30.6
123	DEF	R8 (No Parking)	Left	1st	Left	30.6
456	GHI	R7-22 (Parking)	Left	1st	Left	51.2

Build Parking Zones

Once the LRS data is joined to the parking signs, we use another custom Python script to accomplish the task of joining signs into one or more parking zones.

Working block-wise, we match sign types, and order them as they ascend on a specific side. If the arrow directions for two signs oppose each other, this creates a parking zone.



Support ID	Sign ID	SignType	ArrowDirection	Street	Side	Measure	Parking Zone
123	ABC	R7-22 (Parking)	Right	1st	Left	30.6	Int-DEF
123	DEF	R8 (No Parking)	Left	1st	Left	30.6	ABC-GHI
456	GHI	R7-22 (Parking)	Left	1st	Left	51.2	ABC-GHI

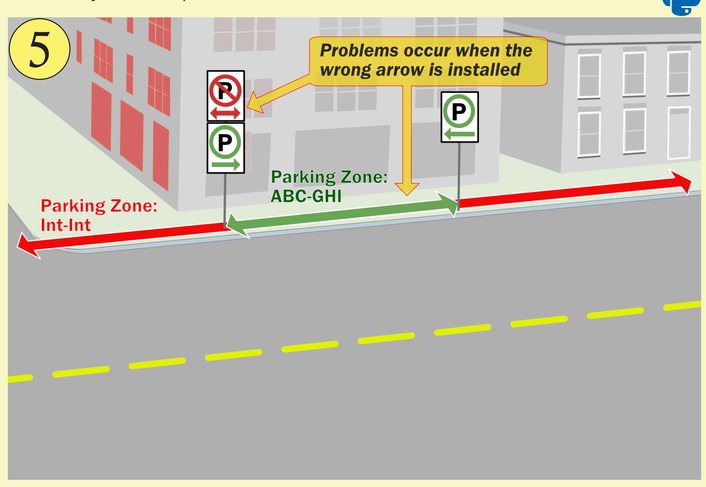
New Parking Zone Table!

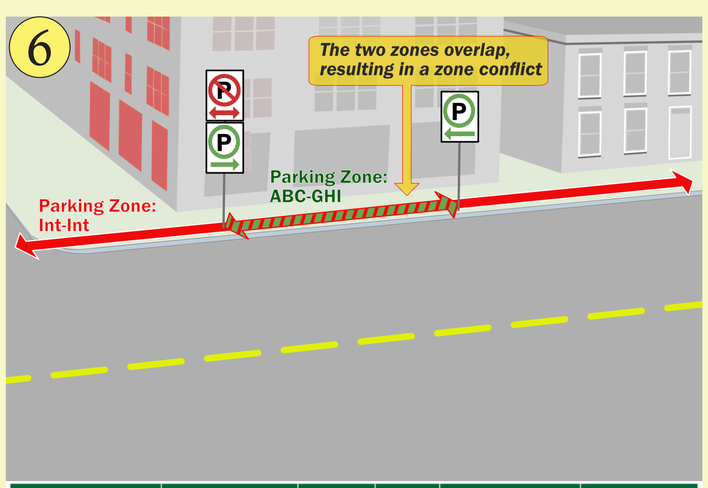
SignType	Parking Zone	Street	Side	FromMeasure	ToMeasure
R8 (No Parking)	Int-DEF	1st	Left	0	30.6
R7-22 (Parking)	ABC-GHI	1st	Left	30.6	51.2

Automatically Sensing Problems

One of the big problems for DDOT has been when the parking sign information is conflicting. Unfortunately, if DDOT doesn't notice it, a resident will (usually after they receive a ticket).

The example below shows when an old sign is replaced (panel 5) with a new one. If the arrow is different, the signs will conflict. (panel 6) However because the sign data have the street, side and measure values, it is quite easy to detect the presence of this overlap with yet another custom Python script.





SignType	Parking Zone	Street	Side	FromMeasure	ToMeasure
R8 (No Parking)	Int-Int	1st	Left	0	99.9
R7-22 (Parking)	ABC-GHI	1st	Left	30.6	51.2

A