**An Examination of OSMNx and Google Maps Features to Improve the HydroXplorer Application**

**Niharika Sri Parasa, Jack Collins**

Last updated: 05-02-2024

**Objective**

Build a pilot[1] exploring different practices for the routing and elevation data, which improve over using only Open Street Map (OSM). The specific cases these techniques aim to improve are shared-pedestrian zones and train tunnels.

**HydroXplorer Application**

The HydroXplorer application is built to serve firefighters in determining the area covered by existing and planned hydrants. Some of the core features of the application include the following:

· Identifying accessible zones utilizing hydrant location or natural water resources.

· Locates nearby water resources for firefighting purposes.

· Calculate elevation disparities between the fire location and surrounding water sources.

So far, the application has been developed using the open-source Python library Open Street Map Network (OSMNx[2]). OSMnx is a Python library to easily download, model, analyze, and visualize street networks and other geospatial features from OpenStreetMap. This library has successfully implemented a majority of the required features. Table 1 illustrates necessary features for the HydroXplorer application and indicates their availability in both OSMNx and Google libraries. (Please refer to the Table 1 below).

However, this case study aims to delve into (a) optimizing the existing features (Table 1 -SNO : 1,2,3) implemented using Open Street Map Network (OSMNx) , (b) check feasibility of few edge cases (Table 1- SNO 4,5,8 ) such as identifying shared pedestrian zones and train tunnels and (c) check any features to improve User Interface/User Experience (Table 1 - SNO : 6,7) with the objective of understanding and implementing these in application.

To accomplish this, we draw a comparison between OSMNx and the Python library for Google Maps. The Python library for Google Maps, also known as `googlemaps`, is a powerful tool that brings the Google Maps Platform Web Services to your Python application[3]. It provides a Python Client library for various Google Maps APIs. It's a community-supported library, and it's designed to be stable and feature-rich enough for building real production applications.

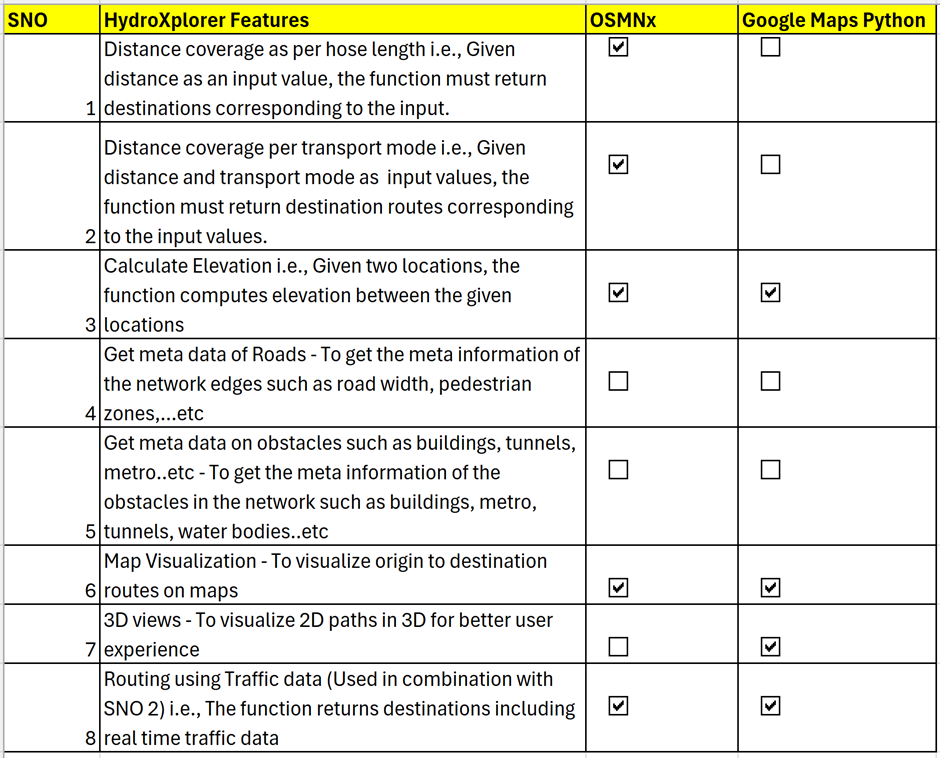


Table 1 : This table offers a comparative analysis of the functionalities available in the Open Street Map Network (OSMNx)[4] and Google Maps Python[5] libraries.

**Observations and Results**

**(a)** Optimizing the existing features (Table 1 - SNO 1,2,3) implemented using OSMNx

The core features “ distance covered per Hose Length, per transport mode and elevation calculation are implemented in the current version of the HydroXplorer application. When comparing features from OSMNx to Google Maps the following was noted:

1. Google Maps Python API does not have any functions to compute routes based on distance i.e, Given a distance value as input, the function should calculate the distance traveled from the origin and return the destinations corresponding to the input distance.

2. Given two locations, Google Maps can calculate elevation. (Implemented in the prototype1).

**(b)** Check feasibility of few edge cases (Table 1 - SNO 4,5,8 ) such as identifying shared pedestrian zones and train tunnels

There is no direct method for features (Table 1 - SNO 4,5,8) in either libraries. So when explored both libraries whether such functionalities exist, the following was noted:

1. OSMNx:

* This library includes methods to retrieve attributes of various entities such as highways and metro lanes, either by name or address. The following image illustrates the attributes provided by the package. However, our specific need to obtain the width of roads is not catered to by the attributes offered by this library.



Source: OSMNx[6]

* Also, OSMNx may not be as current, and the data it possesses could potentially be outdated or no longer relevant. Hence it is not reliable to rely on this feature of OSMNx.
* We explored another alternative OSM\_ROAD\_LENGTH[7] library. This returns lengths of various entities such as roads, sidewalks, metro’s, tunnels…etc. The challenge w.r.t this library is that it doesn’t accommodate road width or any other information on tunnels and pedestrians apart from length. Also, there is no possibility to validate the returned output.(Implemented in pilot1)
* Alternative : We need to utilize graph theory approaches to search, process the road networks along with some meta information from images or so, to implement our requirement.[8]

2. Google Maps Python :

* Google Maps Python has limited functions when compared to OSMNx i.e meta data to get attributes of road or metro tunnels cannot be called via Python API’s. Rather one has to use AI techniques to process satellite images to find road width, mark pedestrian zones..etc. [9]

**(c)** Check features to improve User Interface/User Experience (Table 1 - SNO : 6,7)

The current implementation using OSMNx uses leaflet js, while switching to Google Maps Javascript, Flutter could enhance the user experience of the application.

Having said that, enhancing the UI using Google Maps Javascript API[10] could be expensive[11] (Google Maps API key needs to be purchased), time consuming and resource intensive.

**Key Takeaways**

Based on the observations above, it’s evident that migrating to Google Maps wouldn’t be beneficial as the Python Client of the Google Maps API lacks most of the essential features.

The key elements restricting better functionality for addressing edge cases are:

* Although Google has the data required, the API does not provide it in a format suitable for hydrant range checking.
* Outside of Google, there is no publicly available source for the train tunnels.
* OSMNX and Google often have the same information about whether a given route is traversable by both pedestrians and vehicles (i.e.: shared traffic zones).

Possible alternative improvements are:

* Allowing the user to specify mixed routes (i.e.: assumes the hose-carriers can freely change between driving and walking). This might assist in cases moving emergency vehicles through shared pedestrian zones.
* In cases here OSMNX data shows an area to be impassable, but the user knows it is passable, allow the user to manually draw a line to overwrite that restriction.

**References**

[1] Prototype :<https://colab.research.google.com/drive/1Jdq6FF57fJGHhVfyCdzo2fSPJyz7KCFK#scrollTo=itmZczgdr6iH>

[2] OSMNx : https://osmnx.readthedocs.io/en/stable/

[3] googlemaps pypi - https://pypi.org/project/googlemaps/

[4] OSMNx :<https://osmnx.readthedocs.io/en/stable/index.html>

[5] Google Maps Python Client :<https://googlemaps.github.io/google-maps-services-python/docs/index.html#googlemaps.Client.snap_to_roads>

[6] OSMNx :<https://www.openstreetmap.org/way/1178195267>

[7] OSM\_ROAD\_LENGTH : https://pypi.org/project/osm-road-length/

[8] Sourced from<https://stackoverflow.com/questions/66564479/get-osm-pedestrian-network-with-osmnx-and-delete-roads-if-a-footway-is-along-thi>

[9] Sourced from<https://www.quora.com/How-do-I-find-the-width-of-a-road-on-Google-Maps>

[10] Google Maps Developer Documentation :<https://developers.google.com/maps/documentation>

[11] Google Maps Pricing : https://mapsplatform.google.com/pricing