Extraction and Use of Geometry Data to Obtain 3D Buildings on a Web Map

Josefa Gómez, Abdelhamid Tayebi, Juan Casado
Computer Science Department,
University of Alcalá
Alcalá de Henares, Spain
email: josefa.gomezp@uah.es, hamid.tayebi@uah.es, juan.casado@edu.uah.es

***Abstract*—This work shows a comparison between two different techniques to obtain 3D buildings on a web map. The first one is based on the XYZ Tiles server of OSM Buildings and the second one is based on the Overpass servers of the collaborative project OpenStreetMap. Several simulations have been carried out to analyze their performance. Benefits and limitations of both methods are discussed.**

Keywords- 3D buildings; OSMBuildings; OpenStreetMap.

I. INTRODUCTION

The need of 3D geometries that model real urban environments has been growing in the last few years for different purposes, such as radio network planning, flight simulators, 3D printing, emergency management, simulations for energy consumption, etc. Concretely, the authors are working in this topic because the 3D urban model is required to develop a web application that computes the propagation loss in urban environments [1]. Therefore, a direct application of the present work is the calculation of the coverage in microcells for radio network planning in outdoor environments.

From the point of view of electromagnetism, there are two different techniques to compute propagation loss: empirical and deterministic approaches. Empirical approaches are widely used because they are fast and do not consume lots of resources. However, they are not as accurate as desired. That is the reason why deterministic methods arose. Deterministic methods are normally based on Geometrical Theory of Diffraction / Uniform Theory of Diffraction (GTD/UTD) techniques such as ray tracing. They provide extremely accurate results because they analyze the exact 3D geometrical model of any real environment. However, deterministic approaches have the disadvantages of being time-consuming and requiring lots of computing resources: both memory and processing. Another problem that hinders the application of ray tracing is the dependence of three-dimensional maps of the place under analysis, since it is quite difficult to access this type of information nowadays. In [2], authors propose to use satellite images from Google to recreate 3D buildings. Currently, other options are available like OpenStreetMap database, a collaborative project to create editable and open source maps, launched in June 2004. Everyone can easily contribute by adding new roads, buildings, points of interest like shops, churches, parks, hospitals, etc. A recent study revealed that over 1.2 million nodes and over 130.000 paths are added every day [3]. Especially in urban regions, a lot of 3D building data have been added in the last few years. In [4], OpenStreetMap data were found out to be accurate to model the radio propagation channel for mobile communication systems. Currently, that is possible mainly due to the fact that OpenStreetMap (OSM) Buildings extract OpenStreetMap's tridimensional building data making it accessible but separated from other OpenStreetMap's data. This information can be gathered with Hypertext Transfer Protocol Secure (HTTPS) queries to a REpresentational State Transfer (REST) Application Programming Interface (API) that will provide a JavaScript Object Notation (JSON) formatted highly detailed geometry description of each building in the desired area. This has been possible thanks to OpenStreetMap flexibility that allows users from all around the world to update and modify every detail of their database easily at any point in time.

In [5], it is shown that OpenStreetMap maps are an alternative comparable with three-dimensional maps that include more details for the calculation of the propagation loss by ray tracing. The distribution of signal intensity over both maps greatly looks alike except in high density areas with low signal intensity. It has also been demonstrated that building height only represents a secondary role as long as it is established high enough. [6] shows the development of a localization algorithm based on the fingerprinting technique in which the environment has been modeled with OpenStreetMap's data. In [7], the communication channel is characterized in an urban environment through ray tracing simulations using OpenStreetMap to reconstruct the 3D model of the environment under analysis. That task was carried out in [8] by using OpenStreetMap and Flickr's data.

In this paper, a comparison between two map services that allow 3D city model generation is studied. It is worthwhile to mention that most of the tools that generate and visualize 3D city models (OSM-3D, OSM Buildings, Glosm, OSM2World, etc.) are based on OpenStreetMap. OpenStreetMap contains global building data (along with many other information). Requests can be made directly to them through Overpass. Most of the Overpass servers have limitations on the number of requests that can be done. The Main Overpass API Instance recommends not to exceed 1000 queries per day and not to download more than 5GB of data per day. Nevertheless, other Overpass servers like Kumi Systems Overpass API are more flexible and do not impose a rate limit. On the other hand, OSM Buildings contains these data from OpenStreetMap and possibly others. Requests can be made to extract data but there are also some limitations to do it. The limitations in this case are greater than in all the previous cases. The API for making requests is a REST API on an XYZ Tiles server, which returns data in GeoJSON (more convenient and concrete than

 ISBN: 978-1-61208-810-5

Overpass). Moreover, OSM Buildings prohibits the massive extraction of data, whereas OpenStreetMap does allow it.

As it can be seen, both methods have their advantages and disadvantages. However, the authors have not found similar works in the literature. A comparison between the two aforementioned methods is shown in Sections II, III and IV. Section II includes some simulations to analyze the response time of OpenStreetMap and Section III presents some simulations to analyze the response time of OSM Buildings. Section IV presents some common problems of both data extraction methods. Finally, conclusions are presented in Section V.

II. EXPERIMENTAL RESULTS USING OPENSTREETMAP

The Overpass API is a read-only API that serves up custom selected parts of the OpenStreetMap map data. It acts as a database over the web: the client sends a query to the API and gets back the data set that corresponds to the query.

Unlike the main API of OpenStreetMap, which is optimized for editing, Overpass API is optimized for data consumers that need a few elements within a glimpse or up to around 10 million elements in some minutes, both selected by search criteria like, e.g., location, type of objects, tag properties, proximity, or combinations of them. It acts as a database backend for various services.

Multiple servers provide access to OpenStreetMap data through this API. Requests can be written in XML language or Overpass Query Language (Overpass QL). In order to use Overpass QL, the server must provide an interpreter route that transforms the query into XML on the server side.

Responses are retrieved on XML by default, but if specified they can also be obtained in JSON, CSV and other less relevant formats. It is worth noticing that JSON responses are not the same as GeoJSON responses. GeoJSON is a well-defined spatial data format based on JSON, which is more general allowing for a more flexible, less structured format. If needed conversion from JSON to GeoJSON could be easily implemented in the client side.

Some Overpass servers provide no restrictions to access OpenStreetMap data. This is a great advantage against other options like OSM Buildings but needs to be exploited carefully. Overpass API allows to specify a timeout option that provides a simple security measure against too long or too complex queries. Nevertheless, there is no protection in the API against too big responses. A response of a big enough area that is fully populated with buildings could easily exceed the maximum heap allocation or the maximum RAM available, which will slow down or crash the process that performed the query.

A valid Overpass QL request that will fetch every building inside an area could be express as:

```
[out:json][timeout:60];(way[building](Y1,X1,Y2,X2);
relation[building][type=multipolygon](Y1,X1,Y2,X2););
out;>;out qt;
```

Executing this query against two of the mentioned overpass servers, Main Overpass API Instance and Kumi Systems Overpass API from which only the first one imposes restrictions to access the OSM data, the following average execution times are obtained. Substituting the variables X1, Y1, X2, Y2 on the previous request, the rectangular area between 58º North, 12º East and 59º North, 13º East has been queried. This are is approximately 9438 km^2 and contains 70683 polygons representing buildings shapes. The following statistics were obtained.

TABLE I. COMPARISON BETWEEN OVERPASS SERVERS

	Kumi Systems Overpass API	**Main Overpass API Instance**
Mean	11s 639ms	12s 119ms
Standard Deviation	3s 412ms	3s 343ms
Maximum	18s 355ms	18s 107ms
Minimum	8s 169ms	7s 263ms

From the presented results in Table I, no relevant performance difference between overpass servers with and without restrictions is found.

III. EXPERIMENTAL RESULTS USING OSM BUILDINGS

One of the main characteristics of OSM Buildings is the use of GeoJSON. GeoJSON is a format for encoding a variety of geographic data structures. It is an open standard format designed for representing simple geographical features, along with their non-spatial attributes. It is based on the JavaScript Object Notation (JSON). Since query outputs already are retrieved in this spatial data standard, there will be no need to perform any transformation on the client side. As long as this is the desired data format to use.

To request data to OSM Buildings, a XYZ Tile API is used. This API divides the earth surface in rectangular regions according to a zoom size. In this case, only a value of 15 as zoom size is available. This is a great interface to iterate over contiguous tiles. It is often used by web maps to access the images that create the background over which spatial data is placed. This API is used in maps like Google Maps, OpenStreetMap or Mapbox.

In an application that calculates propagation over an area using building geometries as input, it is required to precisely delimit the area in which buildings are needed. This cannot be achieved with an XYZ Tile API. This API provides fixed tiles that cover the Earth surface seen from the specified zoom level distance. The tiles are organized in a matrix like grids on a map. Each tile is identified by its zoom level and its two the indexes on the matrix. Due to its specification, if just one small portion of the desired area is inside one of the tiles, the whole tile must be requested. It is not possible to request only a portion of a tile. Filtering the tile data to leave only the data that are also inside the desired area needs to be implemented in the client side.

OSM Buildings impose strong and limiting restrictions to access their data. The characteristics of the restrictions are not clearly specified, in contrast with the transparency on Overpass server restrictions. Experimentally, the limitations found are on the number of requests that can be performed concurrently. This restricts the maximum area to retrieve

 ISBN: 978-1-61208-810-5

concurrently to approximately 100 tiles. Once that limit is reached there is a cool down time, of about two minutes, during no other request from the same IP can be performed.

Data has been retrieved from OSM Buildings and Overpass in the area between 38.7028º North, 9.1955º West and 38.7540º North, 9.1189º West, which is approximately 42 km^2. This area corresponds to 42 tiles of zoom level 15. Comparing the obtained data (see Table II), no relevant mean time differences between Overpass and OSM Buildings are observed. Nevertheless, if the results are analyzed more carefully, the following notes can be made.

TABLE II. COMPARISON BETWEEN OVERPASS AND OSM BUILDINGS

	Kumi Systems Overpass API	**OSM Buildings**
Mean	4s 605ms	3s 9478ms
Standard Deviation	1s 694ms	0s 391ms
Maximum	8s 953ms	4s 7209ms
Minimum	3s 212ms	3s 338ms
Geometries	28.394	25.590

OSM Buildings is a more reliable API in the sense that it presents a minimal standard deviation in the mean request time. The reason for this might be that to retrieve the whole area multiple requests are needed, as many as the number of tiles, which could mitigate deviations on single requests.

On the other hand, Overpass API provides a greater bandwidth for data extraction on a single request. If a request to OSM Buildings had not been performed concurrently, the request time would be greater by a factor of the number of tiles.

Finally, and most importantly, is that the number of extracted geometries in the whole area is different. This has two reasons. OpenStreetMap data are improved daily by users that update, correct and polish. The new buildings added to OpenStreetMap that were not there when OSM Buildings extracted their data will not be in their API until they update them. Also, OSM Buildings filters the data removing overlapping geometries that sometimes appear duplicated on OpenStreetMap.

Taking into account that the disadvantage of both APIs is the missing building height attribute, and that the mean request time for large areas is low enough and similar between them, the selected API is Kumi Systems Overpass API because it presents lower access restrictions.

IV. BUILDING HEIGHT RECONSTRUCTION

Apart from the limitations on data extraction and the performance of the APIs the most important problem that both OpenStreetMap and consequently OSM Buildings face is the absence of the building height. Buildings have different attributes that describe them. The most important is the ground shape of the building. Others are less relevant for an application that calculates propagation, like the color of the building. But some, like the material, the roof shape and specially the building height are quite important. Unfortunately, they are not always present.

To solve this problem multiple actions can be performed. Using the levels building attribute, which denotes the number of building levels that the building has, multiplied by a factor the denotes the building level height is quite effective. The downside of this approach is that this attribute is like the height attribute often missing.

When no attributes of the building can be used to know its height, the solutions found are to use a statistical value like the average or median height of the K nearest buildings surrounding each building with a missing height value. This approach is effective, but it is not ideal if large areas lack of the height building attribute.

On the worst case, and base on the work presented in [6], the missing heights can be substituted by a large enough value without relevantly affecting the final values of the calculated propagation. As an example, Figure 1 shows buildings directly extracted from OpenStreetMap. It is noticeable that many of the buildings lack a height value, so they are presented as plane surfaces. Figure 2 shows those same buildings with their height reconstructed. Both figures have been obtained by using the free and open-source 3D computer graphics software tool Blender [9].

V. CONCLUSIONS

The two APIs presented from which to extract building shapes presented their own strengths and faults. Some of them were shared by both since at the end they use OpenStreetMap data. Overpass was the default access point to use these data but needed extra processing on the client side to transform it to a standard data format like GeoJSON. OSM Buildings, on the other hand, presented strict access limitations that almost prevented to use it.

Since the biggest downside of both APIs (the missing building height attribute) was shared by both of them, and the mean request time for large areas was low enough and similar between them, the selected API was the one with lower access restrictions: Kumi Systems Overpass API.

ACKNOWLEDGMENT

This work is supported by the program “Programa de Estímulo a la Investigación de Jóvenes Investigadores” of Vice rectorate for Research and Knowledge Transfer of the University of Alcala and by the Comunidad de Madrid (Spain) through project CM/JIN/2019-028.

REFERENCES

[1] A. Tayebi, J. Gomez, F. Saez de Adana, O. Gutierrez, and M. Fernandez de Sevilla, "Development of a Web-Based Simulation Tool to Estimate the Path Loss in Outdoor Environments using OpenStreetMaps [Wireless Corner]," IEEE Antennas and Propagation Magazine, vol. 61, no. 1, pp. 123-129, Feb. 2019.

[2] D. He, G. Liang, I. Portilla, and T. Riesgo, "A Novel Method for Radio Propagation Simulation Based on Automatic 3D Environment Reconstruction, "6th European Conference on Antennas and Propagation, Prague, 2012, pp. 1445-1449.

 ISBN: 978-1-61208-810-5

[3] P. Neis and A. Zipf, "Analyzing the Contributor Activity of a Volunteered Geographic Information Project-The Case of OpenStreetMap," iSPRS International Journal of Geo-Information, vol.1, no. 2, pp. 146-165, 2012.

[4] J. Nuckelt, D. M. Rose, T. Jansen, and T. S. Kurner, “On the Use of OpenStreetMap Data for V2X Channel Modeling in Urban Scenarios,” 7th European Conference on Antennas and Propagation, Gothenburg, 2013, pp. 3984-3988.

[5] T. Hänel, M. Schwamborn, A. Bothe, and N. Aschenbruck, "On the map accuracy required for network simulations based on ray launching," 16th International Symposium on World of Wireless, Mobile and Multimedia Networks, Boston, MA, 2015, pp. 1-8.

[6] Z. Dai, R. J. Watson, and P. R Shepherd, “A Propagation Modeling Approach to Urban Navigation”, 11th European Conference on Antennas and Propagation, 2017, pp. 1847-1851.

[7] L. Wang, et al., “Vehicle-to-Infrastructure Channel Characterization in Urban Environment at 28 GHz”, China Communications , vol. 16, no. 2, pp. 36-48, Feb. 2019.

[8] H. Fan and A. Zipf, “Modelling the world in 3D from VGI/Crowdsourced data”. In: Capineri, C, Haklay, M, Huang, H, Antoniou, V, Kettunen, J, Ostermann, F and Purves,R. European Handbook of Crowdsourced Geographic Information, pp. 435–446. London: Ubiquity Press. 2016.

[9] Blender computer tool. https://www.blender.org/ [retrieved: September, 2020]

Figure 1. OpenStreetMap raw building data.

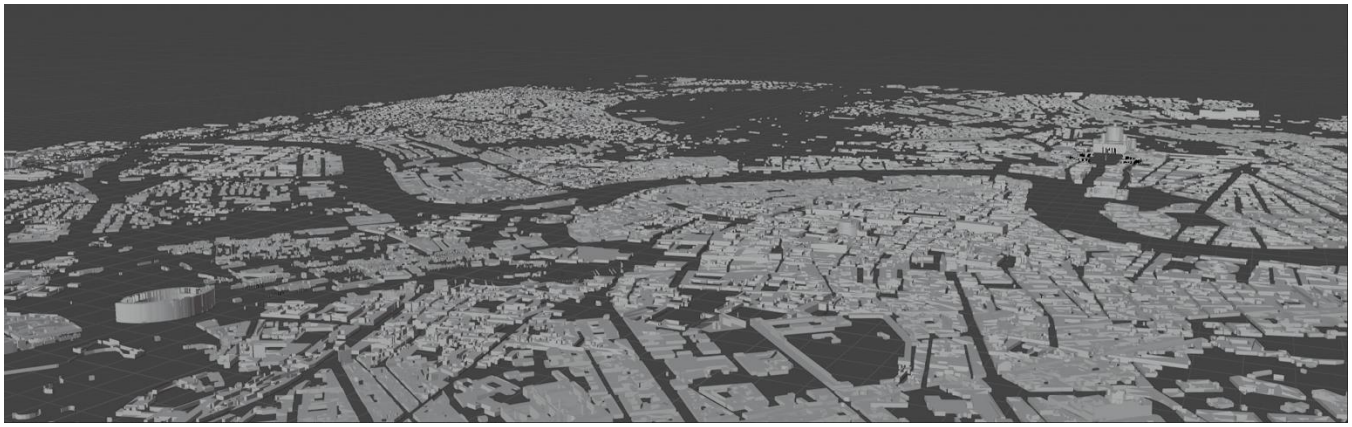

Figure 2. OpenStreetMap building data with infered building height.