Automated 3D Urban Landscapes Visualization Using Open Data Sources on the Example of the City of Zagreb

Adrian Komadina[1] · **Željka Mihajlović**[1]

Received: 30 December 2021 / Accepted: 1 March 2022

Abstract
Geographical location visualization is important to create virtual environments when performing simulations. This study uses publicly available data sources to create three-dimensional (3D) views of geographical locations, focusing on the city of Zagreb as a case study, and presents an automated solution that integrates data from four different sources to achieve this. Publicly available data sources (OpenStreetMap, Google Maps Elevation API, MapBox Static Images API) were used, and GIS Zrinjevac which is unique to Zagreb. Based on these sources, individual 3D objects were generated and displayed virtually in the form of an interactive 3D map. The proposed solution attempts to balance the differing levels of detail achieved using other techniques. The solution was implemented in the form of an application created using the Unity game engine. The results were analyzed to evaluate the solution's validity and the created application's performance. The analysis showed that the total execution time dependence is quadratically polynomial to the amount of retrieved data, and linear to the number of height map points. Examining the application's update rate showed that the buildings and individual models used for polygon and point data had the greatest impact. Finally, possible improvements, alternative approaches, and advantages and disadvantages of the proposed solution are compared with other techniques used in this research area.

Keywords Data visualization · Cartographic data · Virtual environment · OpenStreetMap · GIS Zrinjevac · Google maps elevation API

✉ Adrian Komadina
adrian.komadina@fer.hr

✉ Željka Mihajlović
zeljka.mihajlovic@fer.hr

[1] Department of Electronics, Microelectronics, Computer and Intelligent Systems, Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia

Automatisierte 3D-Visualisierung urbaner Landschaften unter Verwendung offener Datenquellen am Beispiel der Stadt Zagreb

Zusammenfassung
Die Visualisierung geographischer Standorte ist wichtig, um virtuelle Umgebungen zur Durchführung von Simulationen zu erstellen. Diese Studie verwendet öffentlich verfügbare Datenquellen, um dreidimensionale (3D) Ansichten geographischer Standorte abzuleiten, wobei der Schwerpunkt auf der Stadt Zagreb als Fallstudie liegt. Die Studie liefert eine automatisierte Lösung, um Daten aus vier verschiedenen Quellen zu integrieren. Verwendet wurden öffentlich verfügbare Datenquellen (OpenStreetMap, Google Maps Elevation API, MapBox Static Images API) sowie GIS Zrinjevac, welches nur in Zagreb verfügbar ist. Auf der Grundlage dieser Quellen wurden individuelle 3D-Objekte generiert und in einer interaktiven 3D-Karte virtuell dargestellt. Die vorgeschlagene Lösung versucht, die unterschiedlichen Level of Details auszugleichen, die bei Verwendung anderer Techniken erreicht werden. Die Lösung wurde in Form einer Anwendung implementiert, die mit der Game-Engine Unity erstellt wurde. Um die Lösung im Ganzen fachgerecht zu bewerten sowie die Leistungsfähigkeit der entwickelten Anwendung zu überprüfen, wurden die Ergebnisse analysiert. Dabei zeigte sich, dass die Ausführungszeit im Ganzen quadratisch-polynomial abhängig ist von der Menge der abgerufenen Daten. Zudem ist sie linear abhängig zur Anzahl der Höhenkartenpunkte. Bei der Untersuchung der Aktualisierungsrate der Anwendung zeigte sich, dass die Gebäude sowie die einzelnen Modelle, die für Polygon- und Punktdaten verwendet wurden, den größten Einfluss hatten. Am Ende werden mögliche Verbesserungen, alternative Ansätze sowie die Vor- und Nachteile der vorgeschlagenen Lösung mit anderen in diesem Forschungssegment eingesetzten Techniken verglichen.

1 Introduction

It is possible to find cartographic data faster and easier today than ever before. Such data are easy to process and display in a two-dimensional (2D) space, but displaying this in a three-dimensional (3D) space is a more challenging task as many attributes are insufficient or difficult to visualize. A 2D display is sometimes convenient, but does not often provide a complete picture of the observed area. Figure 1 shows a comparison of map representations for the same area in 2D and 3D. Although current technologies offer numerous data sources, many are not publicly available for download. As a result, the focus of this study is on publicly available data sources, of which OpenStreetMap (OSM) is the most important. Other sources used include Google Maps Elevation API, MapBox Static Images API, and GIS Zrinjevac.

Modeling of 3D areas, especially cities, is an important topic in geomatics and can be achieved using various techniques. Three-dimensional modeling techniques are automatically classified as automatic, semi-automatic and manual, and also by data input methods based on laser scanning and photogrammetry (Singh et al. 2013a). A model making use of aerial photogrammetry is currently one of the most popular in the visualization of cities, and it is possible to create a semi-automatic model of city visualization using such datasets (Buyukdemircioglu et al. 2018). In addition

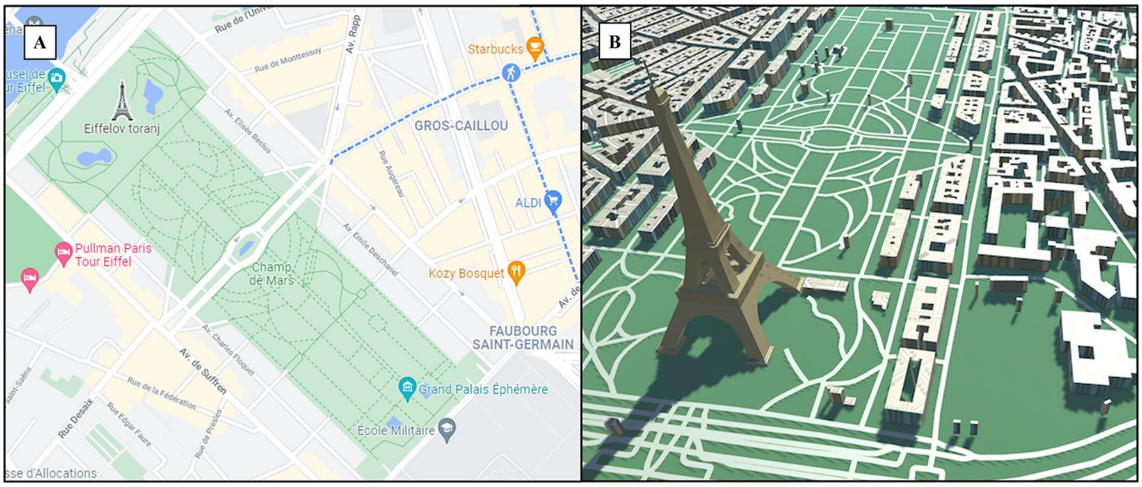

Fig. 1 Comparison of (**A**) 2D (source: https://www.google.com/maps) and (**B**) 3D map representations of the urban area of Paris (source: https://developers.google.cn/maps/documentation/gaming/overview_musk?hl=zh-cn)

to aerial photogrammetry, satellite photogrammetry is also commonly used (Kocaman et al. 2006) to provide examples of modeling 3D cities from high-quality satellite imagery using various software tools. Although it is possible to create high-quality city models using satellite and aerial imagery, these images are often not accessible to everyone. Therefore, a method of close-range photogrammetry has been developed, in which a 3D model of the city can be created from a series of images taken with an ordinary handheld digital camera (Singh et al. 2013b) or using a recorded video and extracting the necessary images from it (Singh et al. 2014).

In addition to photogrammetry, laser scanning may be used, either from air or from the ground. Terrestrial laser scanners can be used for automated 3D modeling of urban environments (Bostrom et al. 2006), and light detection and ranging (LiDAR) technology has been shown to produce photorealistic virtual representations of large cities at relatively low costs (Jovanović et al. 2020). However, the main drawbacks of LiDAR are the large amount of data obtained by this technique, which needs to be analyzed and processed (Zhou and Neumann 2008), and the limited resolution of the samples near the edges of the surfaces (Verma et al. 2006). LiDAR also performs poorly under certain weather conditions, such as high cloud cover and heavy rain. Additionally, good results are not obtained in areas with highly reflective surfaces (Verma et al. 2006). In contrast, LiDAR's advantages are the fast acquisition of data with high precision, direct measurement of the depth component of objects (Verma et al. 2006) and independence from daylight. Because the image obtained by LiDAR contains many artifacts, this technique is primarily used in cases where there is emphasis on photorealistic representation or aesthetic impression; polygonal model representation is used where the visualization is created only as a basis for further simulations. In such situations, geometric inaccuracies are undesirable. Because the techniques mentioned above involve certain visualization problems, to improve and increase the accuracy of the representation itself, combinations of the described techniques are most often used today, which we also call hybrid methods. Thus, it is possible to combine aerial images with cadastral maps to increase the accuracy of the created model (Flamanc et al. 2003) or to combine aerial images, LiDAR, and total stations to obtain a 3D model of the city (Yang and Lee 2019).

CityGML is one of the most popular data formats used for city visualization, and is designed to source and exchange data to visualize urban areas (Ohori et al. 2015, 2018). In this format, cities are characterized by their level of detail (LOD); although a higher level of detail is better for creating a view, it should be noted that most applications and models only use the LOD1 level of detail. Based on this, the implementation in this study will reach the LOD2 level of detail. Because we know that the CityGML format is very complex and CityGML data are difficult to find, this study uses the OSM data format. This format can be converted to the CityGML detail formats LOD1 and LOD2 relatively easily (Goetz 2013).

When it comes to a data source itself, such as OSM, it is important to note that the data obtained may be incomplete and/or incorrect because of the way it is collected. This is because many contributors do not adhere to recommendations on attributes for objects given during participation, either due to insufficient knowledge or a lack of interest (Stančić et al. 2014). Because the OSM database is incomplete, many studies have assessed the quality of OSM data in specific areas (Fan et al. 2014; Girres and Touya 2010). The accuracy and completeness of OSM data, when only buildings are considered, for the Republic of Croatia varies from place to place. In the city of Zagreb, the accuracy of the city center data is close to 100%, whereas on the city edges, it is closer to 50% (Stančić et al. 2014). By combining the OSM dataset with other sources, it is possible to eliminate problems such as insufficient or incorrect elevation data of objects; for example, elevation data can be found via satellite imagery (Girindran et al. 2020).

The applications of visualization can be divided into non-visualization cases (where the creation of 3D models and the performance of 3D spatial operations are not required) and visualization cases (where the 3D model itself is a very important component of the use case and its use would not make sense without it) (Biljecki et al. 2015). The modeling of cities is most encountered in the context of 3D geospatial location visualization because it is the most challenging, and, accordingly, attracts the most attention from researchers and the most concrete applications for such research can be found. The use of 3D city models created by different visualization techniques can be divided into four categories: planning and design, infrastructure and facilities management, commerce and marketing, and promotion and learning of information about cities (Shiode 2000).

Concrete applications of landscape visualization in three dimensions can be found in the development of 3D navigation systems (Sharkawi et al. 2008), urban land management in the form of reducing land consumption (Ross 2011), and urban development planning as a tool that allows city planners to easily insert proposed buildings into existing 3D environments (Isikdag and Zlatanova 2009) in the simulation of flooding in coastal areas to promote awareness among residents of the dangers of such situations (Yang 2016). In Biljecki et al. (2015), 29 different cases were identified where 3D city models would be useful.

This study seeks to use data available from selected sources to create the most accurate representation of any geographic area in three dimensions. First, the data were retrieved and processed. Each extracted piece of information was accompanied by a specific graphical representation with a description of the advantages and disadvantages of such a representation. The territory representation was implemented using the Unity engine, which in turn used a scripting system based on the C# programming language and publicly available object models. The developed application contained a simple user interface through which it is possible to enter geolocation parameters in the form of latitude and longitude values. After displaying the resulting visualization, the user could move around the created 3D map.

This study is divided into several sections, of which Sect. 2 provides an overview of existing work and progress in the geomatics field. Section 3 describes the data sources used in more detail and considers their availability and impact on the visualization of the selected area. It also provides insights into the geographic area that the researchers primarily considered when undertaking this study. In turn, Sect. 4 specifically describes how the mapped data was represented and the visualization issues the researchers encountered. Section 5 uses statistical methods to show the impact of different data groups and datasets on the performance of the proposed application. A final overview of the topic and the implementation achieved in this study is given in Sect. 6, along with suggestions for possible improvements and future work.

2 Related Work

The introduction of this study refers to work in the area of 3D surface modeling without specifically addressing the researchers' topic or implementation proposed in this paper. This section looks at several research papers that use OSM as their main data source.

Not all works examined are oriented towards the generation of an entire city visualization, and some focus only on a part of a city. Wang and Zipf (2017) focused on the visualization of buildings over OSM, more specifically using the extended IndoorOSM, achieving the LOD4 level of detail as per the CityGML standard. According to Nuhn et al. (2012), the focus of this work is even more specific, as it only investigates the generation of landmarks; essentially, it examines which features classify a building as a landmark and shows how these attributes can be extracted from the OSM data.

Existing research has also focused on visualization creation for the purpose of simulations. For example, Dallmeyer et al. (2013) demonstrated that traffic simulations could be performed using OSM maps. In addition, Hadimlioglu and King (2019) describe a simple visualization procedure to simulate water extent in an area in the case of flooding. For the purpose of this study, the authors limited themselves to the LOD0 and LOD1 levels of detail within the CityGML specification. Further, in addition to the OSM data itself, elevation information obtained via the Shuttle Radar Topography Mission (SRTM) global map was used, and visualization implemented with OpenGL.

Many studies have also used elevation information from other sources in combination with OSM data. For example, Over et al. (2010) used publicly available elevation information provided by the SRTM. The suitability and quality of the OSM data were discussed in Over et al.'s research, in addition to generation. Girindran et al. (2020) also used open satellite elevation data to describe a methodology for generating low-cost 3D city models. Another work that uses exclusively developed routines and open-source libraries to visualize virtual 3D city models in real time is Singla and Padia (2021), which uses imagery from high-resolution satellites and high-resolution digital elevation models, in addition to OSM.

However, Joling's (2017) work most closely matches this study, in terms of topic and implementation. Joling describes the process of visualizing geographic locations with OSM using the Unity drive. In addition to OSM, a global SRTMv3 elevation map is used by Joling, while the terrain is textured using the tool Maperitive.

Over the years, many 3D terrain visualization tools have been developed that can be run in a web browser, such as OSMBuildings (https://osmbuildings.org), which displays only buildings, and F4Maps (https://demo.f4map.com), which produces a visualization down to the level of LOD2 detail, which is also the target of this work. However, the latter tool does not have information on height; therefore, the visualization itself is simplified. The tool OSM2World (http://osm2world.org) has a similar level of detail, but still has the same drawbacks as the previously mentioned tool. Desktop solutions are also available, such as JOSM (https://josm.openstreetmap.de), but these only provide a 2D representation of a map. In addition to the aforementioned solutions, there are software development packages (SDKs) for the Unity engine that use datasets for visualization. The most well known among them is that from Google, the Maps SDK for Unity.

Compared with other research in this area, this study will focus on the trade-off between photorealism and abstraction, that is, the lack of a large amount of detail. Figure 2 (below) shows a comparison between a city visualization obtained by aerial photogrammetry (Buyukdemircioglu et al. 2018) and a visualization produced by the web tool OSMBuildings. The image shows that the use of photogrammetry and laser scanning techniques introduces certain instabilities in terms of object geometry, but also produces visual improvements in contrast to a polygonal representation of the data. One would never be able to store and display as much information in a polygonal representation, as is available through other techniques. That said, a geometrically accurate model is sometimes more important than the details. This is evident in cases where the created representation was later used in

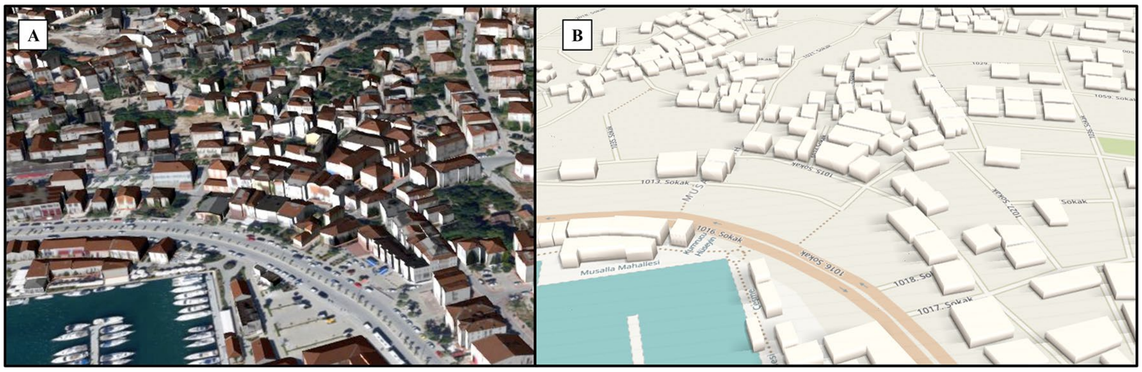

Fig. 2 Comparison of the visualization of the city obtained by (**A**) aerial photogrammetry (Buyukdemircioglu et al., 2018) and (**B**) OSMBuildings web tool (Source: https://osmbuildings.org)

this study as a model for event simulation, where accurate simulation calculations are required. A compromise between the described approaches would be useful in the representation of urban areas, to later perform geographic and urban planning analysis or navigation displays in situations where accurate route calculations are needed, as well as a visually effective representation for a better user experience.

3 Data Used and Study Area

The basis of any 3D map is the surface on which details are to be represented. The terrain is usually not simply a flat surface but depends on the elevation map of a particular area. In this study, information about the elevation map was obtained from the Google Maps elevation API. This simple application interface was used to retrieve information about the elevation of a location on the Earth's surface, as elevation data are available for almost all locations, including those in seas and oceans where the elevation is negative. All elevation values obtained through this service were expressed in meters (m) with respect to the local mean sea level. Working with the described application interface was manifested by sending an HTTP request in the URL that contains information about the location under examination for an elevation value. In addition to elevation, it is desirable to texture the terrain with a satellite image, and details that were not available from other data sources were covered by satellite texture. The MapBox Static Images API, supported by the MapBox GL server, was used to retrieve satellite images.

Major usage restrictions and unavailability of map data around the world have led to the creation of OpenStreetMap. The OSM Project was created in 2004 with the primary goal of creating a public and free map of the world that also had the ability to edit it (Girres and Touya 2010). Today, OSM has more than seven million users who work together to improve the overall system (Madubedube et al. 2021). These users help to improve the OSM using aerial imagery, GPS devices, and simple maps to maintain and validate an accurate and up-to-date cartographic database. The collected information is publicly and freely accessible, which was the main reason that OSM was chosen as the primary data source for this study. OSM uses a topological data structure composed of four basic elements:

- **Node:** a point with geographic coordinates intended to indicate a feature without size.
- **Way:** an ordered list of nodes that may represent a linear or polygonal object.
- **Relation:** an ordered list of nodes, paths, or relations whose purpose is to represent the relationship between the existing nodes and paths.
- **Tag:** a tag is a key-value pair used to store metadata about objects on a map.

Map data can be downloaded for an area in the form of an.osm file and an XML file with a limited set of tags.

Finally, it should be mentioned that an alternative to this method of retrieving OSM data is to use already static files. However, a major drawback of this approach is that it is necessary to know in advance the country in which the area that must be visualized is located. However, if this information is available, one would not need to retrieve or process the OSM data every time. Subsequently, this file could be easily processed using some of the available command tools, such as Osmosis, to extract a specific subset of information.

Although OSM provides a wide range of information, from which data can be extracted for 3D representation, it is quite general; as a result, the amount of information it provides for some areas is small. A data source called GIS Zrinjevac (https://gis.zrinjevac.hr) was therefore used to visualize the Zagreb area. This additional data source represents the green space cadastre of the city of Zagreb and contains information on three types of data: shrubs, trees, and public urban equipment (e.g., benches, tables, trash cans, fountains, playground equipment). GIS Zrinjevac allows its users to view the data on an interactive 2D map; however, to use this data, one must find a way to retrieve it. Looking more closely at Internet traffic, it was noted that each new map view sends a specific HTTP request to the server with the type of data to be retrieved. The server's response to this request includes the data structure with the required information.

This study focuses on the capital and largest city of the Republic of Croatia, Zagreb. The city of Zagreb is located near a geographical position of 45° 50' N and 15° 58' E in northwestern Croatia. Zagreb is situated on a predominantly uneven terrain of approximately 641 km^2 with the lowest point being 122 m and the highest point being 1,035 m above sea level. The average altitude is 158 m, and the estimated population is approximately 800,000, with most buildings located in the lowlands. As we have already described, the additional data source GIS Zrinjevac is available exclusively for this city; therefore, we will base the display and performance evaluation on the geographical area it covers. The visualization is, of course, automated, so it is possible to display any chosen geographical location, but we omit the details included in the mentioned data source.

4 Methodology

Following the described data sources used to implement the visualization, this section describes how the retrieved data were processed and adapted for further use. A brief description of the procedures used to display 2D and 3D elements on the map is also described; the general workflow of the data visualization process described in this section is shown in Fig. 3 below.

4.1 Data Processing and Terrain Generation

A terrain was created as a first step, using the Google Maps Elevation API data source as a basis for displaying additional details within the visualization. The terrain's dimension was determined using a method to calculate the distance between two points in meters, based on their coordinates in the EPSG:4326 system. Because one corner of the terrain aligned with the origin of the scene coordinate system and the terrain extended along the *x*-and *z*-coordinate axes in the positive direction, the increase in the *x*-coordinate represented a greater length and an increase in the *z*-coordinate represented a greater width.

Thereafter, the terrain created was modeled by its elevation data, the information of which was obtained from the Google Elevation API source described earlier. It should be emphasized that it is not possible to specify the elevation for each point of the terrain in Unity, but it is necessary to specify the height map with a predefined resolution in advance; the elevation can then be specified for each point of this map. For the resolution of the height map, one of several predefined options could be chosen (between 33×33 and 4097×4097), where the product was the number of uniformly distributed points in the height map. Thus, it was necessary to send a request to the server for each point in the network. Because the application interface had a limit on the number of locations for which it sent a response, it was not possible to request all points simultaneously. The authors sent a request to the server for each row of the height map; that is, for each latitude increment. Note that the Google Elevation API was not completely free, but its usage was determined by the number of HTTP requests sent. Figure 4 shows a comparison of the created terrain with two different height map resolutions. On the left of Fig. 4, terrain created with a resolution of 65×65 is seen, which had a cost of 65 HTTP requests, while, on the right, terrain with a resolution of 257×257 and a cost of 257 requests is seen.

Another way to obtain the elevation of an area is to use satellite images (Hadimlioglu and King 2019; Joling 2017). The difference in elevation in the downloaded image is determined by the contrast of the image; thus, the lighter parts indicate a higher elevation, and the darker parts indicate a lower elevation. The disadvantage of this method is that the data obtained by these processes are relative, since an absolute value of the height is not available, only its relative ratio. However, its advantage is its independence from Internet access and server services, as well as being able to obtain a continuous bandwidth of altitude information

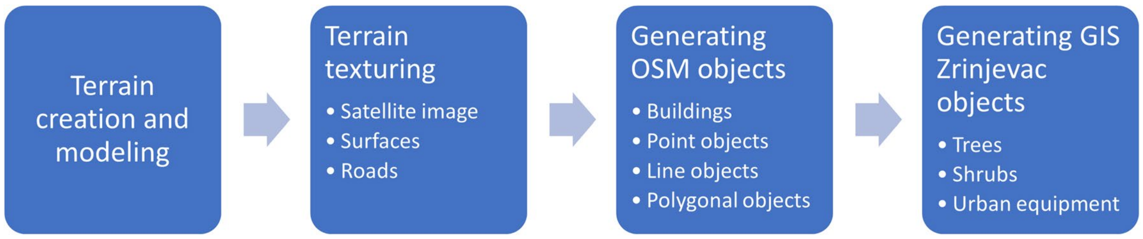

Fig. 3 Flow chart of the general workflow in the data visualization process described in this paper

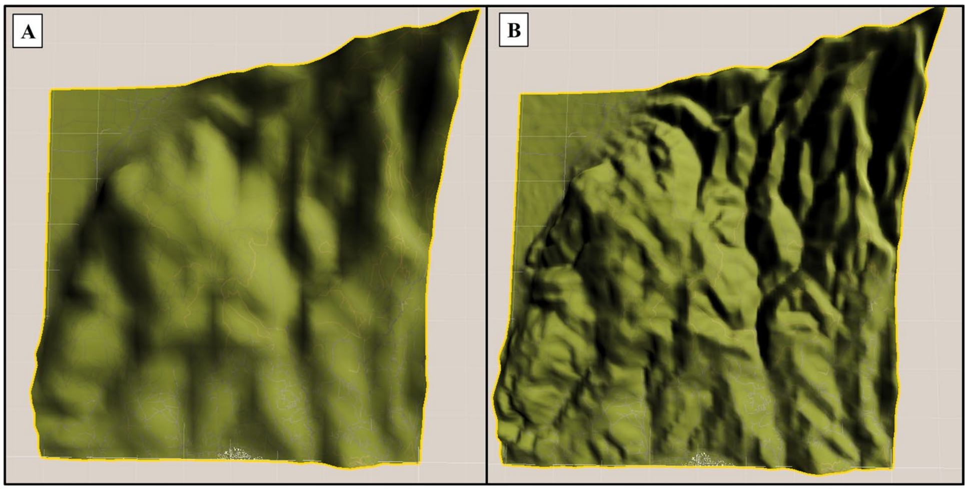

Fig. 4 Comparison of terrain created with **A** 65 × 65 and **B** 257 × 257 heightmap resolution

without sending a large number of HTTP requests, which significantly increases implementation time. As a hybrid method, the use of satellite imagery should be emphasized, with the retrieval of two pieces of information via the Google elevation API to determine the absolute values of all other points in the terrain based on these values and relationships from the imagery.

OSM data are processed using the primitives specified in the OSM dataset description or tag-and-store objects that are displayed in a virtual environment within the C# programming language. In this process, it is necessary to map the retrieved geographic coordinates from the OSM dataset, onto their positions in the scene. This is done by calculating the difference between the retrieved geographic coordinates and the minimum coordinates, and the amount of deviation in the scene space. Data about trees, shrubs, and urban equipment were extracted from the source GIS Zrinjevac. Because this data source was based on the reference coordinate system EPSG:3857, and all other data sources used were based on the system EPSG:4326, it was necessary to convert the edge coordinates to the above system. Locations obtained by calling a method that converts a given location expressed in latitude and longitude to a location in meters were sent to the GIS Zrinjevac server. The location of the data obtained from this source was first stored in the EPSG:3857 system and later converted to the EPSG:4326 system, to be consistent with other data. To obtain additional information about trees and urban equipment for each dataset, an additional HTTP request was sent.

4.2 Data Visualization

The terrain created in this study is a satellite image. To achieve this, it was necessary to texture the terrain using terrain layers. The texture grid was set at a resolution of 4096 × 4096 pixels; this is the maximum resolution allowed in Unity (the higher this value, the more precise the texturing and the more accurate the applied image). The image was simply connected to an appropriate terrain layer and applied to all the points in the texture grid.

The data represented as 2D surfaces on the created terrain are roads, and the data from the OSM dataset represented a surface, such as building land, grassy areas, paved areas, and earth surfaces.

To visualize surfaces, a suitable algorithm was developed, the idea of which was to apply the texture intended to represent this surface to the entire area covered by it. The first step in the execution of the algorithm was to find the positions on the texture grid of all stored reference vertices located inside the path structure. The bounding box was determined from these positions. By traversing all the points of this box, the authors checked whether the point was within the polygon and within the boundaries of the terrain. Based on this, it was determined that the surface layer should be applied to this point in the texture grid. A slightly more complicated texturing procedure related to the visualization of parking lots, where, in addition to the texture of the surface, the white lines that made up the parking lots also had to be placed.

For the algorithm used to generate the roads, the width of the road was first determined based on the number of lanes. In addition, data on the width of a lane were needed, but this data are not available in OSM; this value was therefore estimated depending on the importance of the road. The reference nodes that made up the path structure were unevenly distributed along the path; therefore, the values between them had to be interpolated. Because integer values had to be interpolated between the tiles of the texture grid, the Bresenham algorithm was used. The direction in which part of the road between the two plotted nodes extended was also determined. This determined direction was used to calculate the number of adjacent tiles of the grid that the researchers wanted to texture. For this reason, the determined road width was divided by the length or width of a tile, depending on the axis for which the direction was determined. Finally, alternating road layers of asphalt and white paint were drawn to represent the lines on the road.

One limitation of this approach is the resolution of the texture grid. Because the maximum resolution is limited, it was assumed that for a larger selected area, the area of each tile that made up the grid would increase. Thus, for an area of 1 km × 1 km, one tile covers an area of 0.06 m^2, while for an area of 20 km × 20 km, the same tile covers approximately 24 m^2. Therefore, when representing data by texturing over large areas, necessary accuracy may have been lost. As a possible solution to this problem, the representation of these datasets in 3D was emphasized. While such a representation would provide additional precision in drawing the boundaries of surfaces, it would not provide an excessive visual improvement, because surfaces do not in fact have an oversized 3D component anyway. This would also increase the number of vertices or triangles, which would have affected the performance of the application. For roads, the conclusion was similar, except that it was also necessary to determine their boundary points to visualize in 3D. Road curves could be smoothed in a 3D approach using a curve approximation method, such as the Bezier curve (Joling 2017).

Other OSM data suitable for display as 3D objects in a scene, such as buildings and objects that are point, line, and polygon structures, were represented. The first thing that comes to one's mind when representing a map in 3D is to represent buildings located in an area. The authors' idea was to represent the walls using a mesh in Unity. The authors would apply the appropriate texture to the walls and add doors and, if necessary, windows. Thereafter, a roof was created and add details such as the names of facilities, house numbers, and logos of facilities, added.

To implement the idea described, the height of each building was first determined. If this information was explicitly available in the OSM dataset, it was used; if not, the authors only knew the number of floors and had to approximate the height. This was done according to the following formula: $height = number\ of\ Floors * 3 + 1.5[m]$. The only information available about the building is the position of the polygon vertices that make up the floor plan of the building. To create walls, the positions of the vertices of each wall were calculated. To do this, the authors iterated through all points of the floor plan, taking two adjacent points at a time and duplicating them along the y-axis with an additional offset for the calculated height of the building. The windows were added in a uniform arrangement with rotation in the direction of the wall-normal vector. The procedure for visualizing entrance doors on buildings was similar; however, it was first necessary to determine which wall should be located. This was determined based on the size of the wall and its proximity to the street. When a door was created, representations of other information about the building were similarly created, if known. This information refers to the house number of the building, name of the building, and logo of certain institutions.

The roof-creation algorithm was divided into two cases. In the case where the number of points of the floor plan of the building was equal to four, a roof was created using the information about its shape from the OSM data source. Figure 5 below shows the roof shapes presented as part of the application.

These roof shapes were created by defining all necessary roof points and instantiating quadrangular and triangular meshes based on them. A roof with a plane having more than four points was represented as either flat or pyramidal. A pyramidal roof was selected if the polygon describing the floor plan were convex and if the data source did not explicitly state that the roof was flat. A special method was used for the triangulation of flat roofs because it was difficult to manually create a vertex array for an arbitrarily large number of points. Two cases were distinguished: first, when the polygon undergoing triangulation was convex, and, second, when it was concave. When testing polygon convexity, convex polygons were sometimes declared and those that were not, strictly speaking, mathematically. Most often, these are polygons that contain points forming a straight line; that is, the angle between them is 180°. However, during data processing, there may be slight deviations from this angle, so a small error was allowed. When the polygon representing the roof was convex, a triangulation method called fan triangulation was used. When the polygon representing the roof was concave, triangulation was performed using a downloaded script (http://wiki.unity3d.com/index.php?title=Triangulatoroldid=20279).

The data extracted from the OSM dataset, represented by a node element, are displayed on our map. These were mostly street amenities such as trash cans, ATMs, and fire

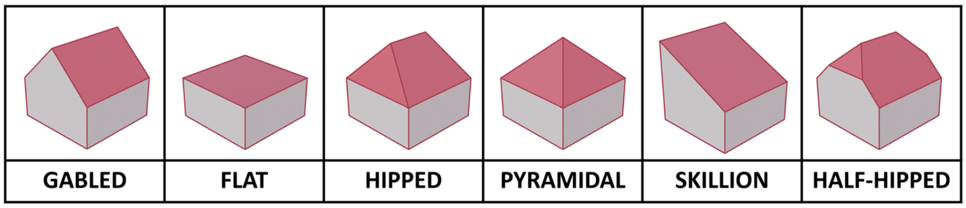

Fig. 5 Types of roof shapes represented in the application from OSM data sources (source: https://wiki.openstreetmap.org/wiki/Key:roof:shape)

hydrants. For this point type of data, only the location and type was available, that is, what it represented. Thus, a corresponding object was created in the virtual scene at the designated position. For some objects, such as one-way street signs, speed limit signs, and street name signs, the exact position was not known, but they were associated with the road. Therefore, the node in the middle of the set of nodes of the associated path was considered the reference location. From this node, the authors searched for the nearest boundary point in the nearby area, where there was no other point object, and created a corresponding object at its position.

The second set of data came from path elements and represented line objects such as fences, railings, and walkways. The process of creating these objects began by calculating the distance between each position of the node on the path. Based on this, and the length of a segment of the object, the number of waypoints that can be found between two nodes was determined. Using this information and calling the Linear interpolation method, a list of waypoints was obtained. Thereafter, a segment of the object oriented was placed along the track at the position of the centroid of each of the two adjacent waypoints.

In addition to the line objects obtained from way structures, polygonal objects were also recognized. However, these were not surfaces that had been visualized in 2D; rather, they were objects whose positions were described by a polygon. The first dataset described sports fields, bowling alleys, canopies, gas stations, greenhouses, and car washes. Objects in the centroid position of the polygon were created, and the direction of rotation was approximated as the longest side of the polygon. If necessary, the object was scaled according to the size of the bounding box of the polygon. Considering data describing bicycle parking, flower fields, or seedling fields, the bounding box for a set of points that describe the polygon were determined. Individual objects were then created at a certain distance, and whether the desired position was within the polygon was checked.

The data visualization from the GIS Zrinjevac data sources was similar to the visualizations described earlier. Structures that have stored tree information contain information about tree height, tree species, and tree location. The models of approximately 20 different tree species were included in the application; if a model to represent a particular tree species was not available, the authors randomly selected a model from the set of those that were available. The process of creating a tree and urban equipment was identical to that of visualizing the OSM point data. Visualization of shrub data was identical to visualization of a field of flowers or seedlings.

Figure 6 shows several screenshots from the application, which in turn shows some of the created objects described in this section. Figure 7 also shows a comparison between the view of the part of the city of Zagreb created using the described method and the actual aerial photographs of the selected area.

5 Results

To analyze the performance and operation of the created application in the context of this study, several different tests were performed targeting different aspects of the application performance. Because the application's operation can be divided into two phases, the analysis of the application was also divided into two phases. The first phase included all parts of the application responsible for data retrieval, data processing, terrain, and object creation, and all the work the application had to do before displaying a 3D map. The second phase included map display and user interaction. All measurements were performed

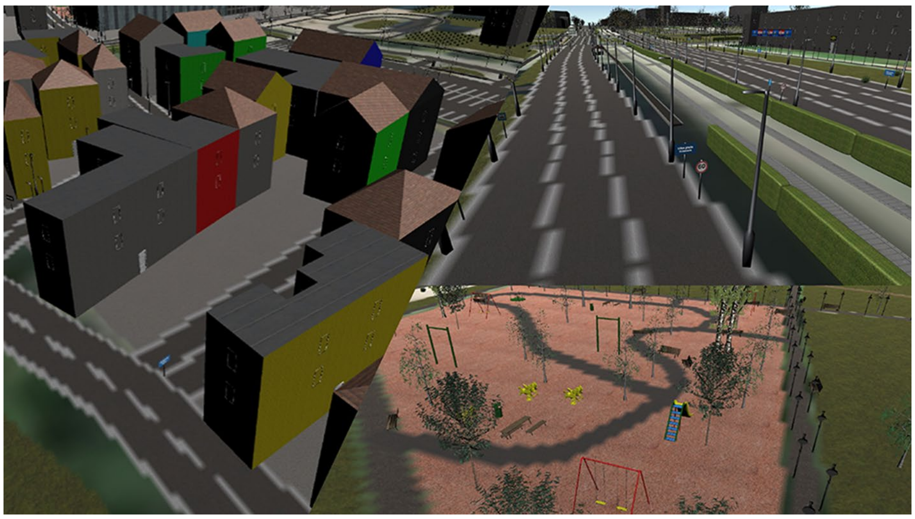

Fig. 6 Screenshot from the created application showing a 3D representation of a part of the city of Zagreb with a focus on buildings, roads, trees and urban equipment

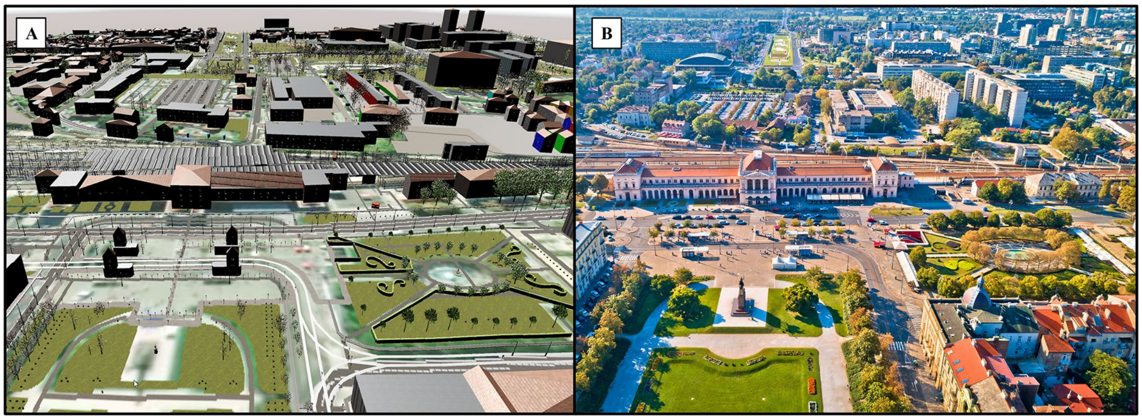

Fig. 7 Comparison of the same location based on (A) a screenshot from the created application, and (B) the actual image of a part of the city of Zagreb (source: https://depositphotos.com)

on an Nvidia GTX 1070 GPU, Intel i7-4770 k quad-core processor with 16 GB RAM.

5.1 Application Execution Time

The authors first examined the time it took to execute all scripts, from the start of the application to the map view and found that they depended on the selected coordinates for which the map was rendered. The selected coordinates primarily determined the size of the terrain itself; however, this did not affect the execution time, which was primarily affected by the amount of data available for the selected area. This set of data included data from the OSM source, GIS Zrinjevac source, and Google Maps Elevation API. The authors concluded that, the larger the volume of data to download and process, the more time it took to do so. How this changes as a function of the amount of data was also investigated. To perform this test, the coordinates (45.81,15.98) were taken as the central point. An increasing range of coordinates were taken relative to the central point, and the time taken from when the request was sent to when the map was rendered on the screen, was recorded. For each measurement, the number of nodes saved was also recorded, including all nodes retrieved from the OSM and GIS Zrinjevac data sources. The number of points on the terrain height map was kept constant at 33×33, the implications of which are discussed later. Figure 8 shows a graph depicting the dependence of the total execution time in seconds on the total number of stored nodes. It also shows the execution times for a total of six measurements, from which it can be seen that the execution time, as a function of the number of nodes, is a polynomial of degree two.

How execution time is related to heightmap resolution was also considered in this study. To this end, measurements for resolutions of 33×33, 65×65, 129×129, and 257×257 were taken. Figure 8 shows a diagram of the

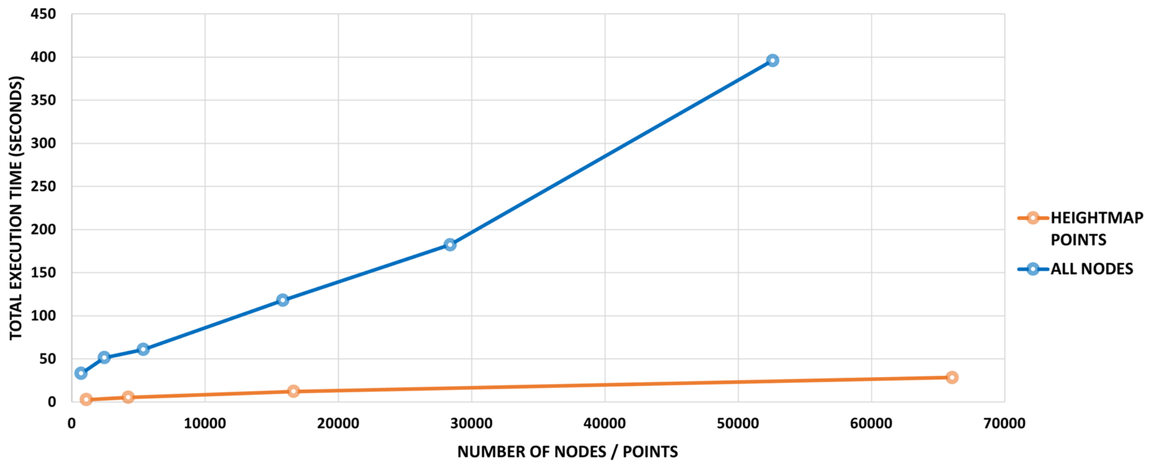

Fig. 8 Dependence of the total execution time of the application on the total number of stored nodes and the number of heightmap points, presented in the form of a connected scatter plot based on the urban area of Zagreb

desired relationship, from which it can be seen that the execution time required to create and set up the terrain is linearly related to the total number of points on the height map.

The percentage of execution time for each step of the application was tested after this. The application's work was divided into six steps: (1) creating and setting up the terrain, (2) retrieving and processing the OSM data, (3) texturing the terrain, (4) visualizing 3D OSM data, and (5) visualizing GIS Zrinjevac data. Figure 9 shows a diagram representing the execution time percentage for each of the above application steps, through six measurements with different node numbers. From the plot shown, it can be concluded that when there are a small number of nodes, the execution time is impacted by the creation, setting, and texturing of the terrain; however, because the time required for these steps does not increase as much as the number of nodes increases, their impact decreases as the number of nodes increases. With a large number of nodes, the biggest impact is the retrieval and processing of OSM data because it is necessary to go through a large amount of OSM data and store all the necessary data in structures. In addition, when the number of nodes is large, the impact of visualizing 3D OSM data increases significantly due to the fact that creating point data often requires calculating the orientation of an object towards the road, which involves traversing all road edge nodes.

5.2 Application Refresh Rate

The next step investigated how individual data affected the application update rate as the user moved around the map. For this measurement, the number of frames per second (FPS) metric was used. The testing method involves positioning the camera so that it is looking at the created map from above, and moving it in certain increments along the *x*- and *z*-axes after each period. Each time the camera was moved, the current number of frames per second was recorded. A total of eight measurements were taken each time, removing a group of data that has a representation in the application. The measurements were taken to create a map that was within the range of [45.8,45.81] latitude and [15.97,15.98] longitude, and the obtained results were presented in boxplots, which can be seen in Fig. 10. From this, it was concluded that line data, shrub data, trees, and urban equipment had no significant impact on the application's operation in terms of frames per second, on average, up to 8%. However, the data with the greatest impact are the building and point data. When building data were removed, the number of frames per second increased by an average of 44%, while without point data, an average increase of 39% FPS was seen.

Figure 11 shows a diagram of the number of triangles and vertices, as a function of the eight groups of data already mentioned. In this diagram, it can be observed that tree models have the largest and most significant number of vertices and triangles in the created virtual scene. It is worth mentioning that the direct view of the camera on the entire displayed scene led to a drastic drop in the frame rate, to only 4 FPS.

Although the measurement results shown in Figs. 10 and 11 appear to contradict each other, the effects of some groups of data on the application performance were different due to the wide variation in how these measurements were taken. Given that the results from Fig. 10 were obtained by measuring the number of frames per second over smaller parts of the scene, and the results from Fig. 11 were obtained using the global view of the scene, the influence of object models (which are relatively

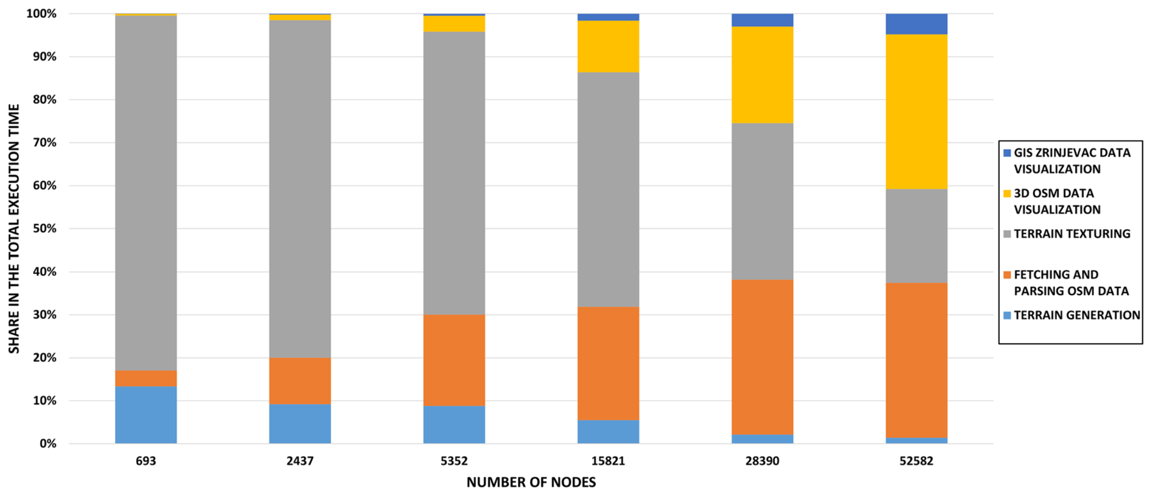

Fig. 9 Proportion of the execution steps of the application in the total execution time in relation to the number of nodes, presented in the form of a stacked column chart based on the urban area of Zagreb

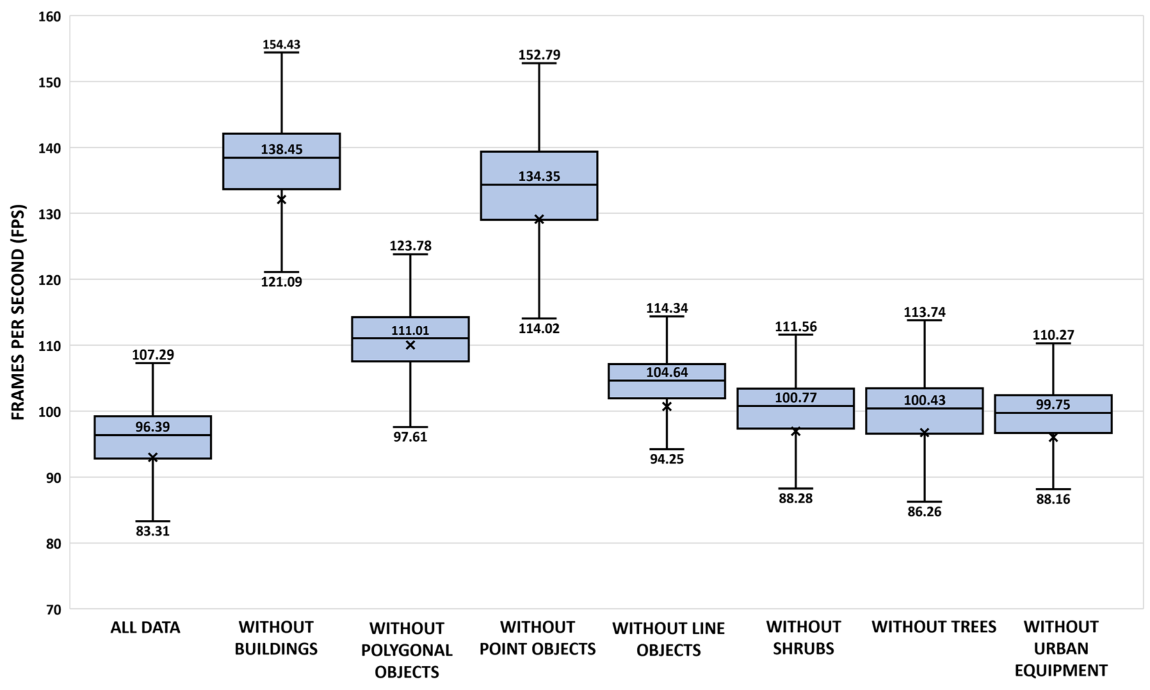

Fig. 10 Dependence of frames per second on a portion of the data used in the application, presented in box plot format, based on the area within the range of [45.8,45.81] latitude and [15.97,15.98] longitude

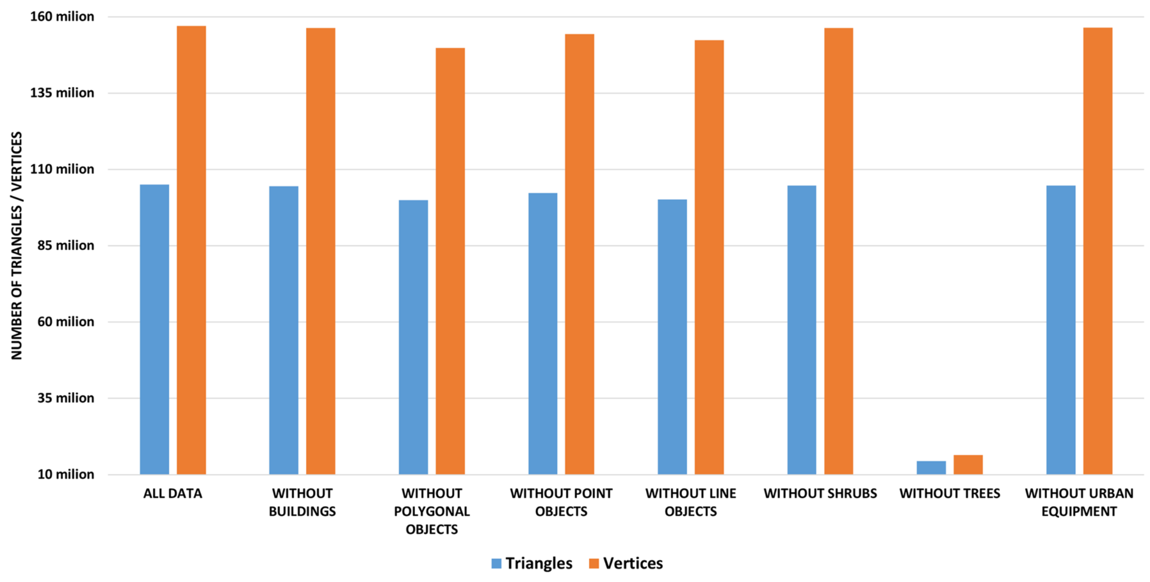

Fig. 11 Dependence of the number of triangles and vertices in the application on a portion of the data used, presented the form of a column chart based on the area within the range of [45.8,45.81] latitude and [15.97,15.98] longitude

rare in the first measurement) was reduced. Consequently, models of trees, which were placed less frequently in the scene, affected the number of frames per second less than models of densely placed buildings. This measurement was set up in this way due to the assumption that a user would focus primarily on only a portion of the map during dynamic review; local models located in the observed area therefore had the greatest impact on their experience.

6 Conclusion

This study presented a new method for visualizing geospatial areas, and used four publicly available data sources to retrieve virtual imagery of real geographies. The retrieved data were processed and displayed as either textures or 3D objects. To implement this proposed new method (and thereby test its relevance), an application was created using the Unity engine to create 3D representations of the area, based on the selected coordinates. The obtained view was in contrast to classic maps, which are also suitable for displaying structures with large height differences (such as mountain ranges), and the data used from the Google Maps Elevation API source came to the fore during the experiment.

The software solution created was used most often to display parts of the city where OSM data dominates; this way, it was possible to distinguish various smaller 3D objects that could not previously be represented using 2D representation techniques. The purpose of the representation obtained can be the static representation of the area as a model or the interactive 3D map for a virtual walk through the represented area. The proposed approach seeks to balance the level of detail between the photorealism and abstraction techniques explored elsewhere in this paper. If the approach relied excessively on photorealism, as is most often the case with photogrammetry and laser scanning techniques, the resulting visualization will possess many geometric inaccuracies; however, while the visual representation would be geometrically accurate, it would not contain much information and detail. It should also be noted that, using the OSM dataset, the visualization is up-to-date with changes in the real world; that is, at each launch of the implementation, new data would be downloaded that reflects the cartographic changes established since the last version.

Examining the execution of this proposed method, it was established that the execution time dependence is, in the worst case, a quadratic polynomial to the amount of retrieved data, and this dependence could be further reduced if the relations of the objects to the roads were not computed. The results also showed that, when viewing a small part of the created area, there were no concerns about the application update rate as there were no problems with the number of frames per second, whereas the update rate dropped significantly when viewing the largest part or the whole area. However, because this view is only interesting from a static point of view, this was not considered a major problem. The authors' analysis showed that the buildings and individual models used for polygon and point data had the greatest impact. Ultimately, the application speed depended on the amount and type of data available for the desired area, which cannot be determined in advance.

Looking to possible future research, further exploring the OpenStreetMap source should be considered, as it contains a large amount of information, not all of which was examined in this study. In particular, an investigation of the relation elements not used in this study would add to the existing research, as it would be likely to contain additional information about the relationships between the path elements and nodes used. This would allow for a more realistic integration of objects into the virtual scene. Additional dimensional information about individual trees could be used in combination with procedural generation to produce a more convincing view. Roads and surfaces could also be represented as 3D objects, which would provide a more accurate representation, although it would require more computing power. To reduce the application's complexity and improve its user experience, it would be beneficial to create a visualization that follows the user's movements, which would require the representation to be rendered piece-by-piece, rather than all at once, at the beginning. This approach is likely to result in the user being able to explore the entire world with a single application launch, rather than specifying a particular area. Further, it should be noted that this approach requires matching the speed of the user's movement with the retrieval and processing of background data. Based on the created map, a component to perform spatial analysis based can be added, as can a traffic simulation or navigation system between geographical points.

Additionally, combining the proposed approach described in this study with other techniques would lead to a hybrid model, which would also provide another area for further research. For example, one possible approach could be to use the OSM dataset to create the geometry of the buildings themselves, and use laser scanning techniques to obtain information about the appearance of these buildings; photogrammetric techniques could then be used to represent the terrain and the actual surface texture at high resolution.

Funding The authors did not receive support from any organization for the submitted work.

Declarations

Conflict of Interest The authors have no competing interests to declare that are relevant to the content of this article.

References

Biljecki F, Stoter J, Ledoux H, Zlatanova S, Çöltekin A (2015) Applications of 3d city models: state of the art review. ISPRS Int J Geo Inf 4(4):2842–2889

Bostrom G, Fiocco M, Gonçalves JGM, Sequeira V (2006) Urban 3d modelling using terrestrial laser scanners. Int Arch Photogram Remote Sens 36:279–284

Buyukdemircioglu M, Kocaman S, Isikdag U (2018) Semiautomatic 3d city model generation from large-format aerial images. ISPRS Int J Geo Inf 7(9):339

Dallmeyer J, Lattner A, Timm I (2013) GIS-based traffic simulation using OSM. Data Mining Geoinform Methods Appl. https://doi.org/10.1007/978-1-4614-7669-6_4

Fan H, Zipf A, Qing Fu, Neis P (2014) Quality assessment for building footprints data on openstreetmap. Int J Geogr Inf Sci 28(4):700–719

Flamanc D, Maillet G, Jibrini H (2003) 3d city models: an operational approach using aerial images and cadastral maps. Int Arch Photogram Remote Sensing Spatial Info Sci 34:53–58

Girindran R, Boyd DS, Rosser J, Vijayan D, Long G, Robinson D (2020) On the reliable generation of 3d city models from open data. Urban Science 4(4):47

Girres J-F, Touya G (2010) Quality assessment of the French openstreetmap dataset. Trans GIS 14(4):435–459

Goetz M (2013) Towards generating highly detailed 3d citygml models from openstreetmap. Int J Geogr Inf Sci 27(5):845–865

Hadimlioglu IA, King SA (2019) City maker: reconstruction of cities from openstreetmap data for environmental visualization and simulations. ISPRS Int J Geo-Info 8(7):298

Isikdag U, Zlatanova S (2009) Interactive modelling of buildings in Google Earth: a 3D tool for Urban Planning, pp 52–70. https://doi.org/10.1007/978-3-642-04791-64.

Joling A (2017) Open data sources for 3d data visualisation-generating 3d worlds based on openstreetmaps data. In VISIGRAPP (3: IVAPP), pp 251–258

Jovanović D, Milovanov S, Ruskovski I, Govedarica M, Sladić D, Radulović A, Pajić V (2020) Building virtual 3d city model for smart cities applications: a case study on campus area of the university of novi sad. ISPRS Int J GeoInform 9(8):476

Kocaman LS, Zhang AG, Poli D (2006) 3d city modeling from high-resolution satellite images. Int Arch Photogram Remote Sens Spat Info Sci. https://doi.org/10.3929/ethz-b-000158058

Madubedube A, Coetzee S, Rautenbach V (2021) A contributor-focused intrinsic quality assessment of openstreetmap in mozambique using unsupervised machine learning. ISPRS Int J Geo-Info 10(3):156

Nuhn E, Reinhardt W, Haske B (2012) Generation of landmarks from 3D city models and OSM data. Proceedings of the AGILE'2012 International Conference on Geographic Information Science, Avignon, France

Ohori K, Biljecki F, Kumar K, Ledoux H, Stoter J (2018) Modeling Cities and Landscapes in 3D with CityGML, pp 199–215. ISBN 978-3-319-92861-6. https://doi.org/10.1007/978-3-319-92862-311

Ohori KA, Ledoux H, Biljecki F, Stoter J (2015) Modeling a 3d city model and its levels of detail as a true 4d model. ISPRS Int J Geo-Info 4(3):1055–1075

Over M et al (2010) Generating web-based 3D city models from OpenStreetMap: the current situation in Germany. Comput Environ Urban Syst 34(6):496–507

Ross L (2011) Virtual 3d city models in urban land management-technologies and applications. Dissertation, Technische Universität Berlin, Fakultät VI - Planen Bauen Umwelt

Sharkawi K, Ujang U, Rahman A (2008) Developing 3D navigation system using 3D game engine. In: Advances Towards 3D GIS, pp 131–140. Penerbit UTM

Shiode N (2000) 3d urban models: recent developments in the digital modelling of urban environments in three-dimensions. GeoJournal 52(3):263–269

Singh SP, Jain K, Mandla VR (2013a) Virtual 3d city modeling: techniques and applications. ISPRS-Int Arch Photogram Remote Sens Spat Info Sci XL:73–91

Singh SP, Jain K, Mandla VR (2013b) Virtual 3d campus modeling by using close range photogrammetry. Am J Civil Eng Architecture 1(6):200–205

Singh SP, Jain K, Mandla VR (2014) A new approach towards image based virtual 3d city modeling by using close range photogrammetry. ISPRS Ann Photogram Remote Sens Spat Info Sci 2(5):329–337

Singla JG, Padia K (2021) A novel approach for generation and visualization of virtual 3D city model using open source libraries. J Indian Soc Remote Sens 49:1239–1244. https://doi.org/10.1007/s12524-020-01191-8

Stančić B, Cetl V, Mađer M (2014) Ispitivanje potencijala dobrovoljnih geoinfomacija na primjeru openstreetmapa u hrvatskoj. Kartografija i Geoinformacije 13(22):48–69

Verma V, Kumar R, Hsu S (2006) 3d building detection and modeling from aerial lidar data. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), vol 2, pp 2213–2220. IEEE

Wang Z, Zipf A (2017) Using openstreetmap data to generate building models with their inner structures for 3d maps. ISPRS Ann Photogram Remote Sens Spat Info Sci 4:411–416

Yang B (2016) Gis based 3-d landscape visualization for promoting citizen's awareness of coastal hazard scenarios in flood prone tourism towns. Appl Geogr 76:85–97

Yang B, Lee J (2019) Improving accuracy of automated 3-d building models for smart cities. Int J Dig Earth 12(2):209–227

Zhou Q-Y, Neumann U (2008) Fast and extensible building modelling from airborne lidar data. In Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems, pp 1–8