International Journal of *Geo-Information*

Article

Accessible Routes Integrating Data from Multiple Sources

Miguel R. Luaces [1,*], Jesús A. Fisteus [2], Luis Sánchez-Fernández [2], Mario Munoz-Organero [2], Jesús Balado [3], Lucía Díaz-Vilariño [3] and Henrique Lorenzo [3]

[1] Database Lab., CITIC, Universidade da Coruña, Elviña, 15071 A Coruña, Spain
[2] Department of Telematic Engineering, Universidad Carlos III de Madrid, 28911 Leganés, Spain; jaf@it.uc3m.es (J.A.F.); luiss@it.uc3m.es (L.S.-F.); munozm@it.uc3m.es (M.M.-O.)
[3] Applied Geotechnologies Research Group, Campus Universitario de Vigo, Universidade de Vigo, CINTECX, As Lagoas, Marcosende, 36310 Vigo, Spain; jbalado@uvigo.es (J.B.); lucia@uvigo.es (L.D.-V.); hlorenzo@uvigo.es (H.L.)
* Correspondence: miguel.luaces@udc.es

Abstract: Providing citizens with the ability to move around in an accessible way is a requirement for all cities today. However, modeling city infrastructures so that accessible routes can be computed is a challenge because it involves collecting information from multiple, large-scale and heterogeneous data sources. In this paper, we propose and validate the architecture of an information system that creates an accessibility data model for cities by ingesting data from different types of sources and provides an application that can be used by people with different abilities to compute accessible routes. The article describes the processes that allow building a network of pedestrian infrastructures from the OpenStreetMap information (i.e., sidewalks and pedestrian crossings), improving the network with information extracted obtained from mobile-sensed LiDAR data (i.e., ramps, steps, and pedestrian crossings), detecting obstacles using volunteered information collected from the hardware sensors of the mobile devices of the citizens (i.e., ramps and steps), and detecting accessibility problems with software sensors in social networks (i.e., Twitter). The information system is validated through its application in a case study in the city of Vigo (Spain).

Keywords: spatial data mining; geospatial NLP; geospatial data fusion; large scale geospatial processing; pedestrian navigation; physical accessibility

Citation: Luaces, M.R.; Fisteus, J.A.; Sánchez-Fernández, L.; Munoz-Organero, M.; Balado, J.; Díaz-Vilariño, L.; Lorenzo, H. Accessible Routes Integrating Data from Multiple Sources. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 7. https://dx.doi.org/10.3390/ijgi10010007

Received: 9 November 2020
Accepted: 21 December 2020
Published: 26 December 2020

1. Introduction

Accessibility is considered a general principle of the United Nations Convention on the Rights of Persons with Disabilities, and State Parties undertake to promote the availability and use of new technologies suitable for persons with disabilities. In particular, information and communications technologies can be used to help people find their way through cities avoiding obstacles and barriers to accessibility. Several research studies have already focused on facilitating navigation for people with diverse abilities in Smart Cities. The study in Reference [1] presents a systematic literature review of some of the technological components which are the building blocks of smart accessibility and a comparison of existing technologies and case studies. The majority of existing solutions provide a mobile app-based routing assistant for outdoor door-to-door journey planning using partial sensed data (mainly single source based). However, a holistic approach should combine software and hardware sensors to dynamically and seamlessly detect barriers affecting people with diverse abilities so that barrier-less pathways may be offered to them. Such an information system requires the integration of data coming from heterogeneous sources such as geographic information, LiDAR point clouds, hardware sensors, social networks, and user-volunteered information.

This paper focuses on filling some of the gaps previously identified for the development of the underneath technology that will allow any smart city to create barrier-less mobility assistants adapted to the specific limitations of each citizen. The technology developed allows the construction of layers of information of urban elements affecting the user mobility in public spaces by obtaining knowledge from open sources available on the Internet, from geomatic sensors (such as LiDAR sensors) and by using techniques from crowd-sensed data from both hardware and software sensors generated or captured by the users of the city themselves, particularly those embedded in the mobile or wearable devices carried by the citizens (e.g., accelerometers, gyroscopes, barometers or light and sound sensors) or traces that people leave on the Internet such as in social networks.

Figure 1 shows the general framework of this paper. The left side enumerates the data sources considered in the paper. OpenStreetMap is used to retrieve geographic information regarding points of interest (e.g., destinations, disabled parking spaces) and the sidewalk network. A software component developed using Java allows the automation of the data capture process using tasks defined using a JSON-based declarative language that combines harvesting tasks and processing tasks defined in SQL. We describe the algorithm of a particular processing task that builds the sidewalk network from the OpenStreetMap road network. LiDAR data from selected locations of the city is captured using mobile LiDAR devices. The point clouds collected are processed by a Python component that refines that sidewalk network previously created (e.g., improving the pedestrian crosses) and detects obstacles on the sidewalks (i.e., ramps and steps). The obstacles detected from LiDAR data are processed asynchronously because the survey teams and the mobile LiDAR devices cannot be permanently deployed. However, the user-volunteered information gathered using a mobile application developed in Java for Android Devices is processed synchronously and in near real-time. The application collects accessibility issues explicitly reported by users, and it uses the mobile sensors of the device (i.e., gyroscope, magnetometer, accelerometer, and gravity sensor) to identify and report obstacles using a convolutional neural network implemented using a Keras model. Finally, a Python component uses the Twitter Streaming API to select relevant Tweets using keywords, and then it uses information from the Tweet and natural language processing techniques to extract the location of the Tweet and report an obstacle. All the information collected is represented in an accessibility data model that describes the pedestrian network, the destinations, the disabled parking spaces, and the accessibility barriers that affect the network. Finally, a progressive web application developed using Vue.js and a Java backend allows the user to compute an accessible route including possible obstacles.

The first contribution of this paper is the architecture of an information system to store and analyse the information in a spatio-temporal database that integrates all the above sources of sensed data to obtain an inventory of elements in the urban environment and accessible to citizens with disabilities. The second contribution of the paper are specific algorithms to build a pedestrian network from open data sources and improve the network using LiDAR sensors, to find accessibility barriers both asynchronously (using LiDAR data and a geomatic approach) and synchronously (using user-volunteered data from mobile sensors and a machine learning approach), and to find user-reported obstacles using Twitter data and natural language processing. The third contribution is the description of a data model and a mobile application that can be used to compute to obtain personalized routes to move across the smart city considering the limited mobility constraints of each individual (the blind, people moving in wheelchairs, people with respiratory limitations, etc.).

This paper is organized as follows—Section 2 is dedicated to present related work. The system architecture is fully presented in Section 3. Section 4 describes the details of the implementation. A case study of the application of our system is described in Section 5. Section 6 is dedicated to gathering the major conclusions of this research paper.

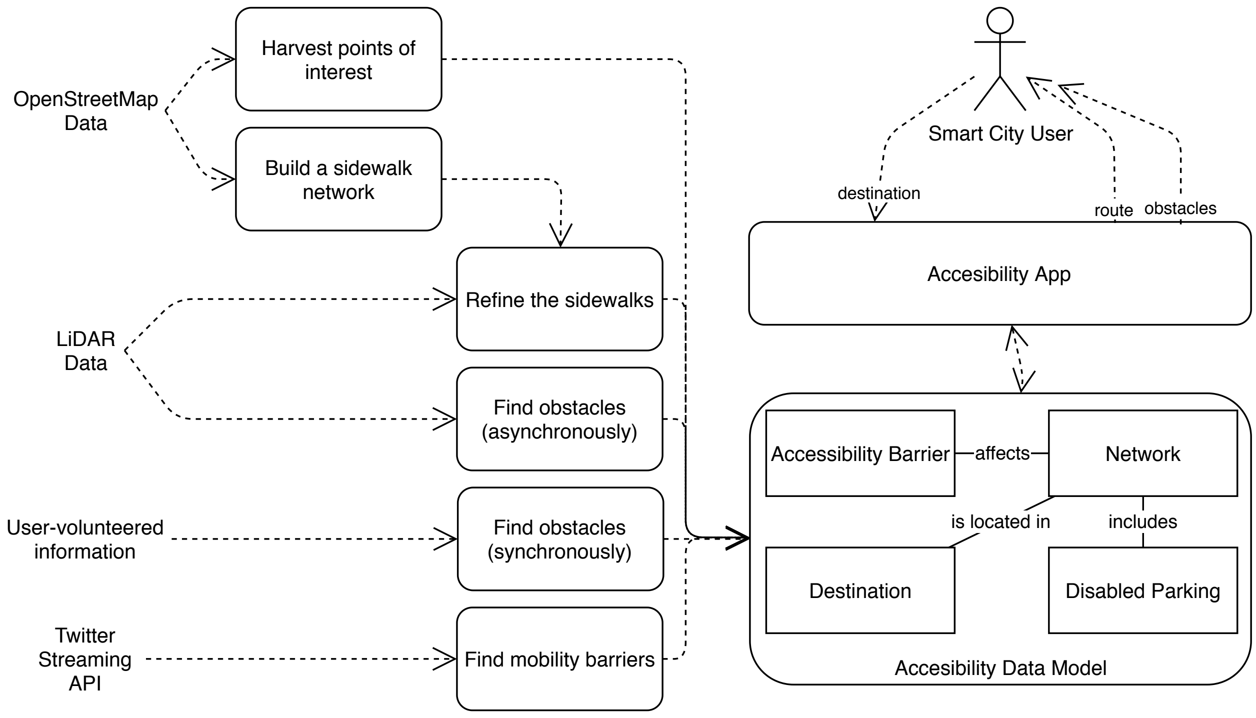


Figure 1. General framework of this paper.

2. Related Work

Recent trends in information and communications technologies have resulted in many different data acquisition technologies and data sources that can provide valuable information to an information system that computes an accessible route in a Smart City. User-volunteered geographic information can provide the foundation of the system, whereas LiDAR sensors and sensors in mobile devices can provide detailed information for the outdoor-indoor frontier. Furthermore, social networks provide a valuable source to gather real-time information regarding accessibility barriers. This section describes related work regarding these data sources.

Collecting data on street-level accessibility is a very complex task. Even tough there are many projects that aim at collecting sidewalk accessibility data [2,3], the current coverage of these tools is limited. OpenStreetMap (OSM) has succeeded in its goal of creating and distributing geographic information for the world. However, OSM is highly-biased to vehicle routing (i.e., road center lines are represented, but not sidewalks) and map visualization (i.e., pedestrian crossings are represented as points). Therefore, until the coverage of publicly-available sources is high enough, in order to build an accessibility data model it is necessary to perform data mining processes on the existing data and define algorithms to build the sidewalk and pedestrian crossing networks from the data collected.

Since physical accessibility depends on the environment geometry, point clouds are a valuable source of information because they represent accurately, in 3D and in real magnitude the environment acquired by a LiDAR equipment. Many works focus on curb detection as elements of accessibility [4,5]. Curbs are easily detected because, in most cases, they are vertical elements connecting two horizontal surfaces (sidewalks and road) and they have a direct view from the Mobile Laser Scanner trajectory. But curbs are not the only important ground elements for physical accessibility. In Reference [6], urban ground elements are segmented and they are classified, based on the geometric and topological features, in sidewalks, roads, curbs, and stairs (divided in risers and threads). There are several methods for detecting stairs based on their geometry as a set of horizontal and vertical surfaces. In the representation of the stairs as histograms, the steps are identifiable as a sequence of regular peaks [7]. Mathematical morphology applied to point clouds also serves to detect steps when one step is used as a structuring element [8]. In addition to detecting the elements, it is necessary to evaluate their condition and dimensions to check their accessibility. In Reference [9], the sidewalks are detected, and their width is measured. In Reference [10], other features as cross slope, grade, and curb ramp slope are extracted on sidewalks and ramps. In Reference [11], the free unobstructed space left by obstacles on the navigable ground is calculated for profiles of people without reduced mobility and people in wheelchairs.

All detection methods improve their effectiveness when the probable location is known beforehand. Some methods focused on the detection of steps and/or ramps indoor entrances [12], building entrances and road-sidewalks intersection [13], and crosswalk environments [14]. Unlike the other elements, zebra crossings are not detectable in point clouds because of their geometry, but because of the reflective paint, which produces peak values in the intensity attribute [15]. Another alternative to the search for geometric forms in the point cloud is through the modeling of the human body. Considering the human body a polyhedron [16] or a model with 41 degrees of freedom [17], it can be moved by the environment to detect obstacles, the steps, the slope of the ground, and the visibility of the signs. In a similar way to a human body, some humanoid robots designed to climb stairs also integrate 3D data capture sensors and software to recognize stairs and calculate the movement needed to climb them [18,19].

The use of sensors in mobile devices is bringing about a revolution in different fields such as environmental sensing, health, activity detection, or social applications [20]. Applied to urban commuting, accelerometry data combined with mobile location allow detection and geolocation of events such as falls [21] and difficulty walking [22]. Light sensors allow the detection of regions of insufficient night lighting. The microphone allows the capture of places with high noise levels. Despite the underlying potential, to date, most work on the use of wearable and wearable sensors for user-centered detection has been based on activity detection (especially those involving periodic movements over time) [23]. Activity detection sometimes leads to the detection of the environment in which these activities are carried out (for example, climbing stairs implies the detection and counting of the number of steps, as well as the ease or difficulty that climbing them represents for the user). However, the potential that the combination of the information captured by the different sensors in mobile devices supposes for the detection of urban elements with an influence on mobility is still to be explored in many cases. On the other hand, mobile sensors (or equivalently sensors in wearable devices such as watches or smart bracelets) allow not only the detection of urban elements but also the detection of the cost for the user to traverse these urban elements in terms of physical and mental burden [24]. Some of the benefits of detection systems based on crowd-sensing using mobile multi-user sensors can be seen in Reference [25], allowing continuous updating of sensed elements with small detection delays.

Several research works can be found in the state of the art that make use of data extracted from social networks to detect real-time events. One of the first and more popular examples is the work of Sakaki et al. [26]. The authors developed a system that collected Twitter data from Japanese Twitter users. They were able to detect 96% of the earthquakes reported by the Japan Meteorological Agency. In many cases, their system was able to report the earthquakes before they were published by the ads of the Meteorology Agency.

In the context of smart cities, we can find several works that exploit Twitter data to detect events. Metro Averías [27] is a system that collects complaints from Twitter users about Madrid Metro, the underground transportation system of Madrid. Besides, the system publishes reports about the most frequent complaints per line and alerts about possible breakdowns in the Twitter account @metroaverias. Several works can be found in the state of the art that develop systems that extract data from social networks to detect traffic events like accidents or traffic jams. Among them, we can mention the works of Anastasi et al. [28] and Kumar et al. [29]. Finally, Wanichayapong et al. [30] developed a system that extracts Twitter data to detect traffic events, but also obstruction hazards and road conditions. They focus their work on the detection of transportation events in roads and highways. They use a dictionary with Places and Verbs written in the Thai language, where Places are names related to roads and their elements and Verbs are terms related to traffic problems. Then they collected tweets that contain at least one term from the Places dictionary and another term from the Verbs dictionary and filtered those tweets that contained profanity or vulgarity terms.

3. System Architecture

Figure 2 shows the architecture of *FlatCity*, the system that we have developed to create a mobility assistant that integrates information from multiple heterogeneous data sources. The figure shows the three main components of the system. The upper left component (i.e., *Ingestion*) represents the data ingestion component that is responsible for collecting data from multiple sources to build the accessibility model of the city. The information collected and processed is stored in the *Accessibility Data Model* component shown at the bottom of the figure. Finally, the upper right component that represents the *Applications* that are offered to users of the system to calculate the most accessible routes.

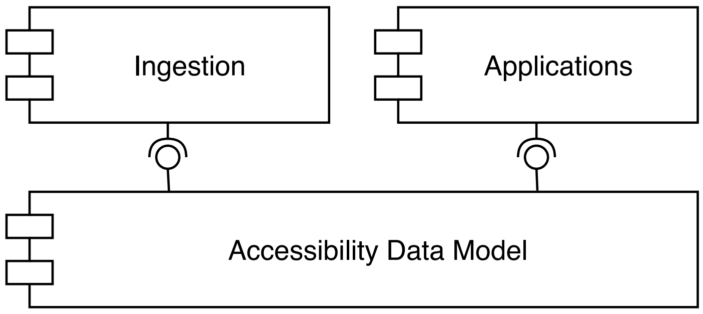


Figure 2. FlatCity System architecture.

Figure 3 shows a UML component diagram of the architecture of the ingestion subsystem of Flatcity. It is a layered architecture with four layers. The top layer (*data sources*) consists of the four data sources that are currently considered in Flatcity: OpenStreetMap, surveying teams using mobile LiDAR scanners, Smart City users providing volunteered crowd-sensed information, and Twitter. The second layer (*data collectors*) consists of the components that are responsible for collecting the information from the data sources and send it to the ingestion back-end. This layer provides the system with independence from the data sources so that it is possible to modify them without altering the ingestion system. Furthermore, as it is organized into small independent components, it is possible to distribute them in cloud computing infrastructures in a simpler way. The third layer (*data ingestion*) is a component that provides REST endpoints to receive the information from the data sources and stores it in the data model. It is in turn structured as a collection of independent services (*Network API*, *Accessibility API*, *Obstacle API*, *Issue API*, and *Social API*, not shown as components in the figure) to support load balancing. The bottom layer is the Flatcity accessibility data model stored in a PostgreSQL database.

The first data source that we consider in Flatcity is OpenStreetMap. The *OSM Harvester* component is in charge of retrieving all the accessibility-related information for the whole area of interest (usually, the entire city) and process it to build the street network and to collect the reduced-mobility parking places, the destinations, and their reduced-mobility entrances. The information collected from OpenStreetMap is used as the baseline information for Flatcity that will be afterward refined with information collected from the other data sources. The *OSM Harvester* component has to deal with the problem that OpenStreetMap data is designed to represent roads but not sidewalks, pedestrian crossings, and other city infrastructures meant for pedestrians and not for vehicles. Therefore, the *OSM Harvester* component has to build a sidewalk network using the information collected from OpenStreetMap. This process is described in further detail in Section 4.1.

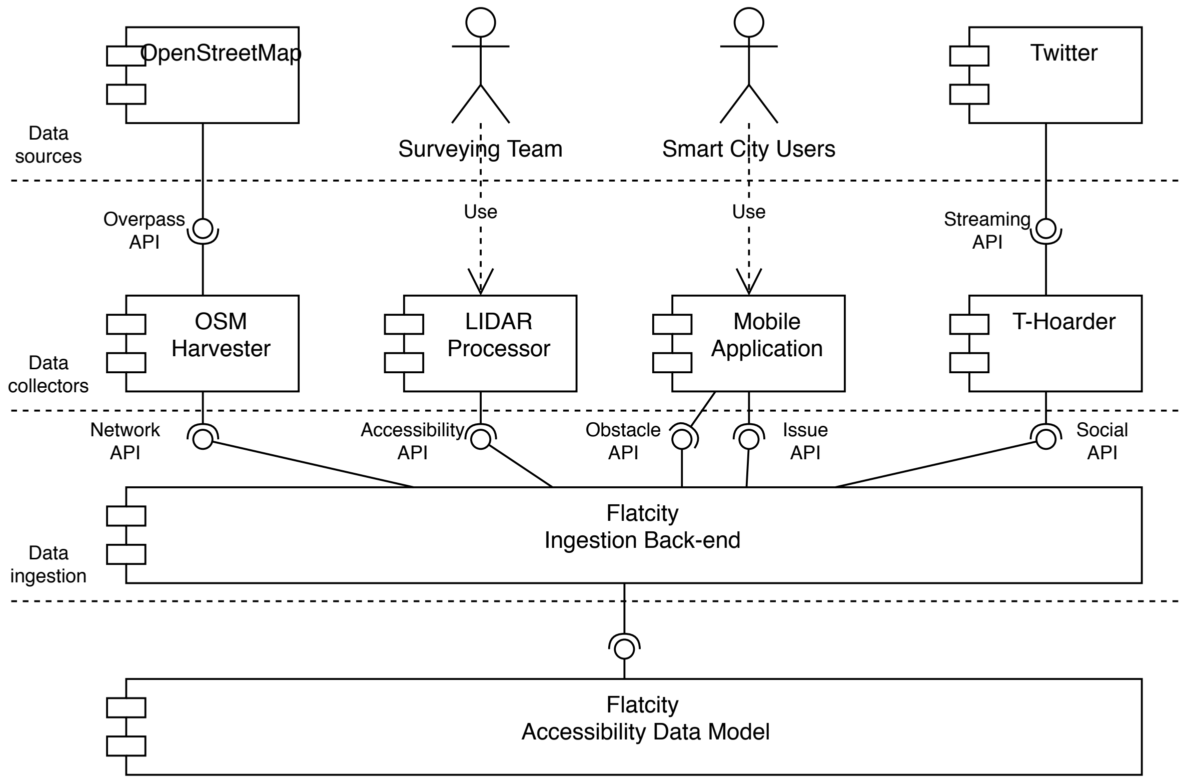


Figure 3. Ingestion architecture.

The second data source is a surveying team using mobile LiDAR to discover accessibility barriers. A team equipped with a mobile laser scanner drive or walk through specific areas of interest and collects a point cloud. The point cloud is processed by the *LiDAR processor* component and detailed information regarding accessibility (i.e., ramps, steps, and pedestrian crossings) is collected. The information collected by this component can be used to improve the baseline information collected from OpenStreetMap in areas of the city where the OpenStreetMap data is poor or in areas of the city where there is a special interest regarding accessibility (e.g., hospitals or other public buildings). The information provided by the *LiDAR Processor* component to the ingestion back-end is much more precise than the information collected from OpenStreetMap. Therefore, the ingestion back-end has to perform a conflation process to integrate the coarse-grained OSM information with the fine-grained LiDAR data. The detection and the conflation processes are described in Section 4.2.

The third data source is a mobile application for Smart City users. Two different types of user-volunteered information can be retrieved. First, the mobile sensors of the device can be used to identify and report obstacles. Second, users may explicitly report accessibility issues. The mobile application collects the information and sends it to the ingestion back-end. This information can be used to provide more detail over the OpenStreetMap data in areas where the LiDAR information cannot be collected (e.g., private areas) or in areas that cannot be covered with the LiDAR information because of the cost of retrieving the information. Also, due to cost constraints, the mobile application can help to keep the information about obstacles updated (we expect that in a real scenario the use of LiDAR devices will be used sparsely). The mobile application is described in Section 4.3.

The fourth data source is the accessibility problems reported on Twitter. A software sensor (named *T-Hoarder*) is used to filter the Twitter Streaming API using relevant keywords. For each Tweet that is received, T-Hoarder extracts the geographic location of the Tweet (either because it is provided by the user or extracted from the text). Then, T-Hoarder reports the problem to the Flatcity ingestion back-end. The component is described with more detail in Section 4.4.

Figure 4 shows the data model managed by the *Accessibility Data Model* component. The *Edge* and *Node* classes at the top of the data model represent the graph of the city's pedestrian infrastructure network. Each *Edge* has two attributes that represent the cost of traveling the edge in both directions (i.e., *normal_cost* and *reverse_cost*). It also has an attribute that represents the level of accessibility of the edge (i.e., *accessibility*). A value of 1 in this attribute indicates that the edge can be traversed by anyone. On the other hand, a value of 0 indicates that it is an edge with barriers that can only be traversed by a non-disabled person. The values between 0 and 1 can be used to represent different levels of barriers in the edge. Finally, the *Edge* class also has an attribute that represents the geographical path of the edge (i.e., *trajectory*).

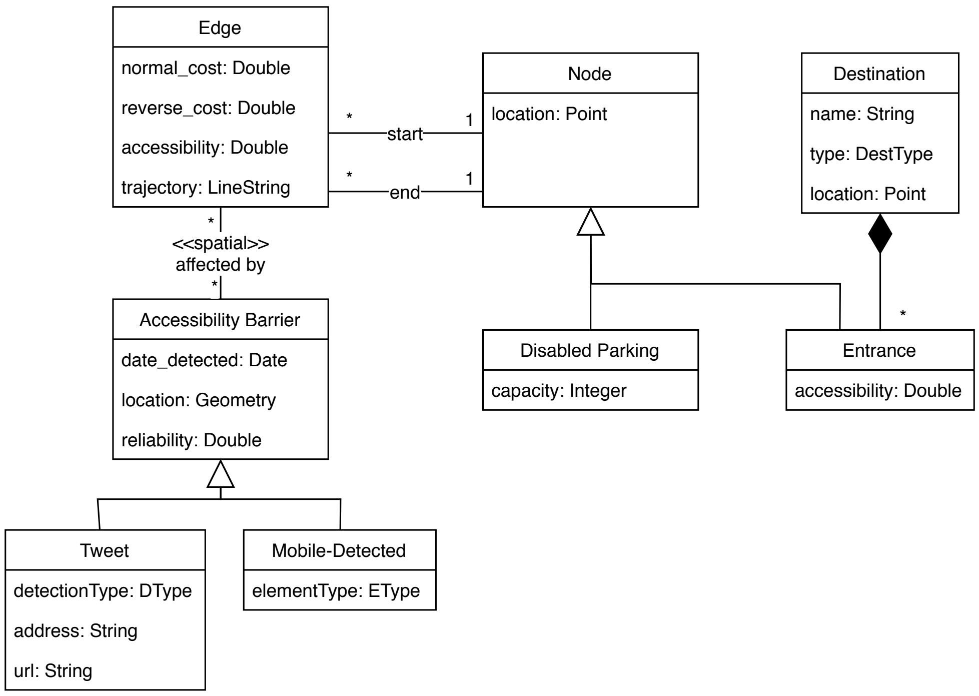


Figure 4. Data model.

The *Accessibility Barrier* class on the left of the data model represents the obstacles that are identified by the mobile application for Smart City users and the Twitter software sensor. The class has an attribute to represent the date on which the detection occurred (i.e., *date_detected*) to ignore obstacles that may be obsolete. The class can also represent the geographical location of the obstacle (i.e., *location*) using the generic *Geometry* data type so that not only points but also lines and surfaces can be used. Finally, the attribute *reliability* represents the confidence of the detection to inform the user of the probability of the obstacles being real. The class *Accessibility Barrier* is specialized into two classes to represent each of the types of obstacles that are currently supported by FlatCity: those detected from Tweets and those detected using the mobile application. The class *Tweet* includes detection type used to extract the geographic location of the Tweet (i.e., geolocated, georeferenced or extracted using natural-language processing; more detail can be found in Section 4.4), the address detected, and the URL of the Tweet. The class *Mobile-Detected* includes only the type of element detected. The *Accessibility Barrier* class is associated to the edges of the network that are affected by the barrier. The association is stereotyped as *«spatial »* to indicate that the edges affected by each barrier can be detected using a spatial query involving the geographic information in both classes.

The classes that represent the possible places that the user can indicate as the destination of the route are shown on the right side of the data model. On the one hand, the *Disabled Parking* class represents the mobility-reduced parking places available in the city. The attribute *capacity* can be used to indicate that there is more than one parking place in a single location. On the other hand, the *Destination* class represents the possible points of interest that can be used as the end point of a route. The class includes attributes to represent the name of the destination, the type of destination, and the geographic location. Furthermore, each destination includes a set of entrances (represented by the *Entrance* class). This allows destinations that are large buildings with multiple entrances to be adequately represented. Besides, each entrance includes the *accessibility* attribute with the same semantics that were described for the edges. Therefore, the difficulty of using each particular entrance of a large building can be described.

Figure 5 shows the architecture of the applications that can be used by the Smart City users to compute accessible routes. Both applications use the *Accessibility Data Model* through a REST endpoint that is implemented in the *Accessibility Back-end* component. Two different applications are provided—a web application and a mobile application. Both applications provide functionality to find a route from a given origin to a destination (selected from the destinations represented on the data model, or clicking on the map). In the general case, the route is computed in three sections—from the origin to the location of the car of the user, from the car to the mobility-reduced parking place closest to the destination, and from the parking place to the entrance of the destination. The user is expected to walk in the first and the third sections, and therefore the routes are computed using the *Accessibility Data Model*. The middle section is expected to be traveled by car, and therefore the *Accessibility Back-end* delegates the computation of this section to an external route provider (e.g., GraphHopper (https://www.graphhopper.com/), Open Source Routing Machine (http://project-osrm.org/), or the Google Directions API). To allow the application to compute the route in these three sections, it provides functionality to remember the place where the car was parked. Furthermore, the user can also store the location of its home place to use it as origin or destination of the route.

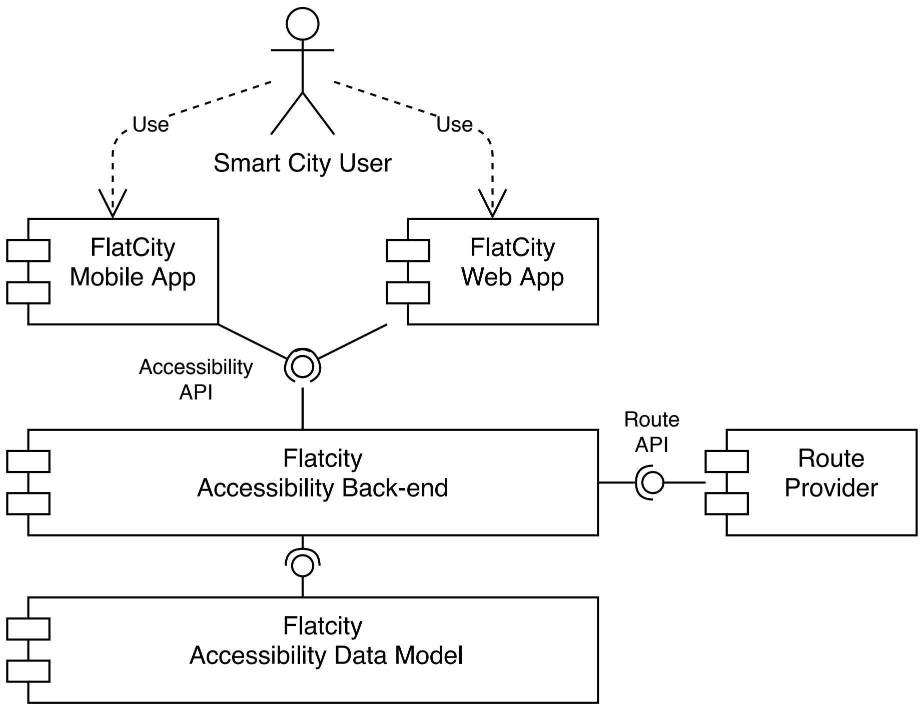


Figure 5. Application architecture.

4. Implementation

4.1. Extracting Information from OpenStreetMap

4.1.1. Automation of the Data Capture Process

The *OSM Harvester* component has been designed to allow automation of the Open-SteetMap data capture process. It allows the definition of *harvesting tasks* whose conceptual model is shown in Figure 6. Each *Task* is composed of three elements. The first one is the *Configuration* of the task that consist of the *connection* information to the target database, the *target schema* into which the data must be copied, and the *action* to be performed. The action can be one of the following:

- **Create**. Creates the schema from scratch before importing OSM data.
- **Update**. Deletes all data from the schema before importing new OSM data.
- **Append**. Appends new OSM data to the existent in the schema.
- **Import.** Runs the import scripts without harvesting data from OSM.

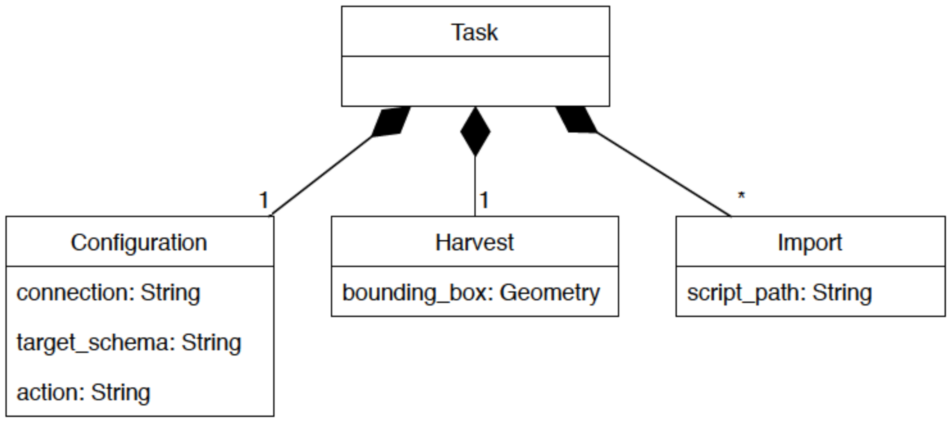


Figure 6. Conceptual model of an *OSM Harverster* task.

The second element of the task is the *harvest definition*. It currently consists of the *bounding box* of the area of interest, but we plan to include filters in the future. We use the Overpass API (https://overpass-api.de/) to extract the OSM data and *osmosis* (https://github.com/openstreetmap/osmosis) to parse and import the OSM data into the database. The third element of the task is a collection of *import scripts* that must be executed after the data is harvested. We currently support SQL scripts, but we plan to include support for other scripting languages in the future. The process that builds the network described in the next section is implemented as an SQL script. Listing 1 shows an example of an *OSM Harverster* script that imports the banks, pharmacies and supermarkets into the table of destinations.

Listing 1: Example of an *OSM Harverster* script.

```
INSERT INTO t_destination (id, location, name, type)
SELECT id, geom AS location, tags >'name' AS name,
CASE WHEN tags >'shop' = 'supermarket' THEN 'supermarket'
ELSE tags >'amenity'
END AS type
FROM osm.nodes
WHERE tags >'amenity' IN ('bank', 'pharmacy')
OR tags >'shop' IN ('supermarket');
```

4.1.2. Building the Network

OpenStreetMap is designed to describe the infrastructure of roads for vehicles. It is possible to describe the sidewalks, but there are only about 1.8 million of sidewalk segments (1,805,845 segments as of 5th November 2020 as seen at https://wiki.openstreetmap.org/wiki/Key:sidewalk) in the OpenStreetMap database compared to over 170 million road segments (170,086,925 segments as of 5th November 2020 as seen at https://wiki.openstreetmap.org/wiki/Key:highway). Furthermore pedestrian crossings in OpenStreetMap are being tagged as points in the road section instead of lines between sidewalks (3,069,077 points compared to 822,689 lines as of 5th November 2020 as seen at https://wiki.openstreetmap.org/wiki/Key:crossing). Hence, to build a network that can be used to compute accessible routes, we have designed and algorithm to build the network of sidewalks and pedestrian crossings.

The algorithm that we have designed builds first a network of sidewalks, and then creates pedestrian crossings between the sidewalks. To build the network of sidewalks we take into account three cases:

- The road segments tagged as *steps* in OpenStreetMap are included directly with the accessibility value set to 0 (i.e., can only be used by non-disabled persons).
- The road segments tagged as *footway*, *pedestrian*, *path*, or *track* in OpenStreetMap are included directly with the accessibility value set to 1 (i.e., can only be used by non-disabled persons). As future work, we plan to perform an analysis of the relief to find difficult ramps.
- We build a buffer of 4 m to each side of all the other road segments that are not tagged as *motorway* in OpenStreetMap. Then, we aggregate all the road buffers to build the geometry of *paved area* and we extract the boundary of the paved area as a sidewalk. Figure 7 shows an example of this process. The dark blue lines represent the road segments extracted from OpenStreetMap. The light blue surfaces represent the paved area built using a 4m buffer around the roads. The black lines represent the sidewalks computed as the boundary of the paved areas. Hence, we assume that there is a sidewalk to each side of each road that is not restricted to pedestrians.

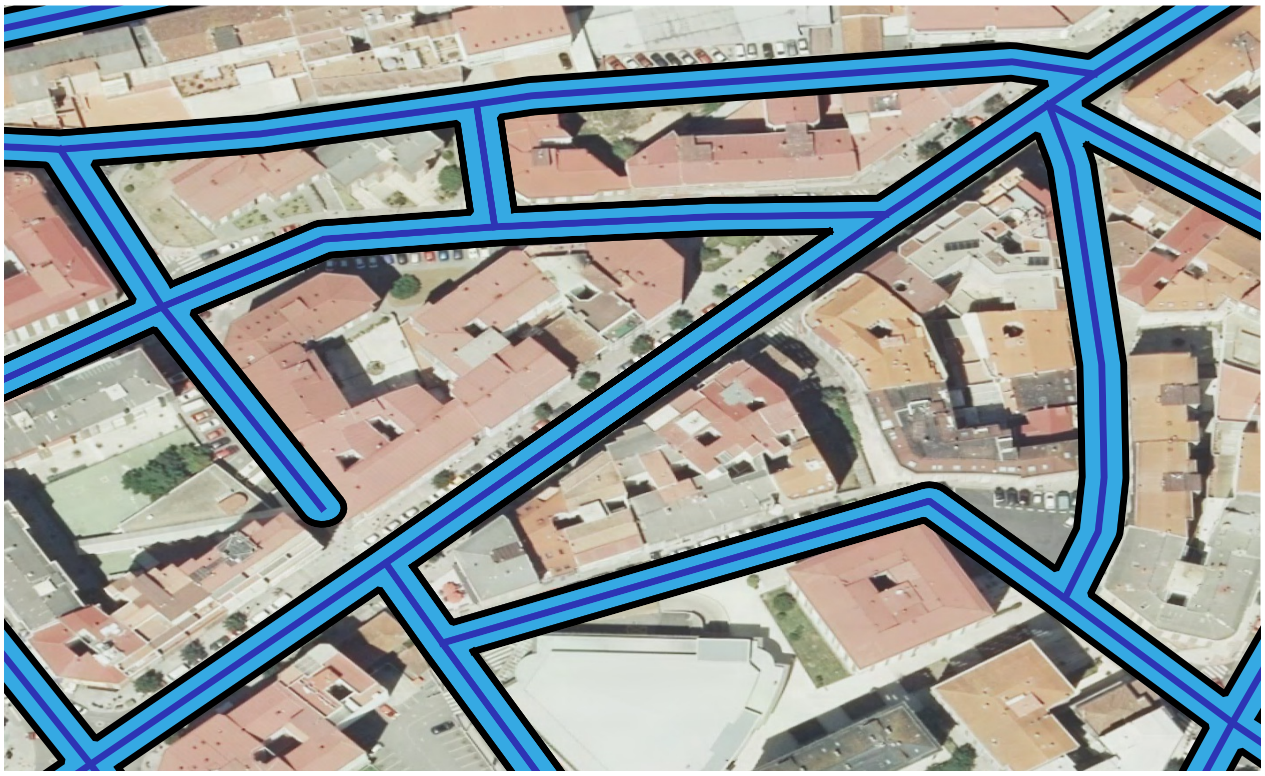


Figure 7. Building a network of sidewalks from OpenStreetMap (OSM) roads.

This algorithm assumes that there is a sidewalk on both sides of the road. It is impossible to verify whether this holds using only OpenStreetMap data. However, it is possible to verify it using LiDAR data and remove sidewalks that do not exist.

To add pedestrian crossings to the network, we first build a straight line from each crossing to each sidewalk section that is closer than 8 m. This creates a star-like pattern from each crossing to the sidewalks. Then, we remove all the line segments that have another line segment that is shorter and with a difference in orientation smaller than 45 degrees. The remaining line segments are considered the pedestrian crossings. Figure 8 shows an example of this process. The light dark blue points represent the crossings extracted from OpenStreetMap. The light blue lines represent the star-like pattern created from each crossing to each sidewalk segment closer than 8 m. Finally, the green lines represent the pedestrian crossings that are left after removing line segments. The result is a simpler network of pedestrian crossings that makes the route computation more efficient even though there are still has additional pedestrian crossings that are not real.

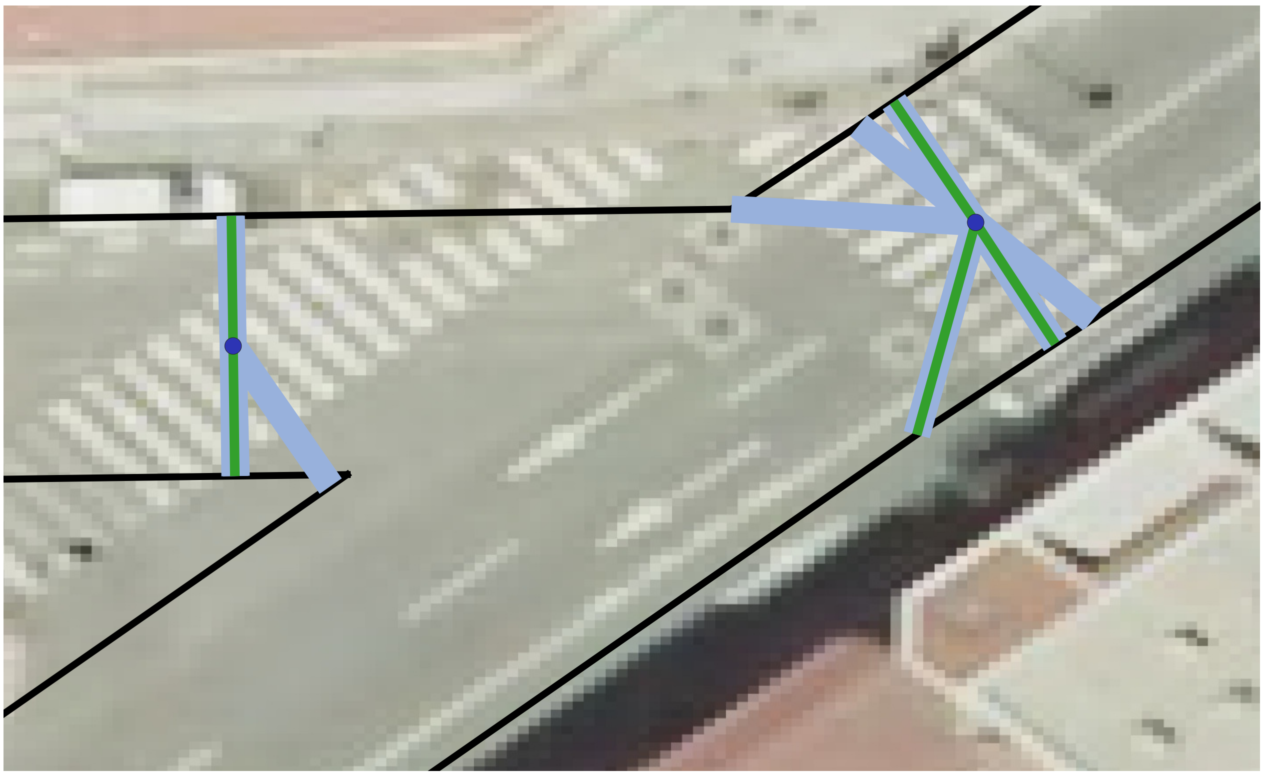


Figure 8. Building pedestrian crossings from OSM crossings.

4.2. Network and Obstacles Detection from LiDAR Data

4.2.1. Detection

The elements forming the urban ground are automatically segmented and classified by the method proposed in Reference [13]. The input data is a point cloud of an urban street with one line of facades. First, the area near the ground plane of the point cloud is delimited as a region of interest (ROI) and detected based on the RANSAC algorithm. Then, the surface normals of each point are calculated with respect to its nearest neighbors in order to know the inclination of each point. The vertical points are rasterized to calculate the height of the vertical elements and to detect the steps and curbs, whose height is less than h. Finally, given the inclination of each point and the topological relations with other ground elements, the ground points are segmented and classified into roads, curbs or curb-ramps, sidewalks, and steps or ramps.

The detection of crosswalks, as the other main element in pedestrian navigation along the sidewalks, is based on the method proposed in Reference [15]. The points corresponding to crosswalks are recognizable because of the intensity attribute, as the reflective paint generates intensity peaks. When high-intensity points are detected, a bounding box is generated that groups the entire crosswalk as a single element.

4.2.2. Modeling

The modeling process of each ground element is modified from the method proposed in Reference [31]: modeling of each element separately and based on its size. Sidewalks are modeled based on their length locating each node separated to generate a pedestrian map. The number of nodes of each sidewalk section is obtained from the length of each section divided by d. Then, the k-means algorithm is applied to the sidewalk points group the points around each future node. Each node is located in the geometric center of each group of points. Finally, once located the sidewalk nodes, the edges between the sidewalk nodes are estimated based on the search for other nodes at a distance d.

The ramps and steps of the building entrances are modeled as one node located in the geometric center of each element. Subsequently, each ramp or step node is connected to the nearest sidewalk node. The crosswalks are modeled as one node at the geometric center and connected to the nearest sidewalk node. If the edge that connects the crosswalk with the sidewalk crosses a set of points belonging to a curb or curb-ramps, the edge is replaced by two edges, and an intermediate node (step or ramp class as appropriate) is generated connecting with the sidewalk and the crosswalk nodes. Curbs and roads are not modeled, as they are not pedestrian navigation elements. In this way, a pedestrian map is generated based on sidewalks with accessibility information on building entrances and crosswalks.

4.2.3. Integration of the LiDAR Model in the OSM Model

Given a sidewalk and crosswalk map obtained from OSM, and the more accurate map generated from LiDAR data in the previous subsections, the objective of this process is to integrate the LiDAR map into the OSM, replacing the corresponding edges of the OSM map. On the OSM map, the sidewalks are defined as lines and the crosswalks as polylines. The integration process of the pavements and entrances is shown in Figure 9. First, a buffer B is generated around the sidewalk elements of the LiDAR map to look for the edges of the OSM map that must be replaced. Edges or nodes of building entrances are not used for the buffer generation. The lines of the OSM map to be substituted are those that intersect with buffer B. The connection nodes Cn of the OSM map with the LiDAR map are detected as the end-point of the OSM line that crosses the border of B and is outside the buffer B. From the points Cn, the nearest nodes Ln are searched in the LiDAR map. Then, new lines are generated whose beginning-end points are defined by Cn and Ln, replacing the lines crossing the border of B. In case a sidewalk is not detected in the point cloud, the corresponding lines of the OSM map would be deleted.

The procedure is similar for pedestrian crossing integration. Given as input data a point cloud of one line of facades, in the LiDAR model, only half crosswalks are represented, defined by the central point and the connection with the nearest sidewalks through a curb or ramp nodes. A buffer is generated around crosswalk elements modeled from LiDAR data to detect the polylines of the OSM intersecting. The OSM polyline is replaced by a line defined by: (1) the node of the curb (or curb-ramp) connecting crosswalk with LiDAR sidewalk model and (2) the node connecting OSM crosswalk with the front OSM sidewalk. This point is detected by searching for the furthest point of the polyline OSM crosswalk to LiDAR sidewalk connection.

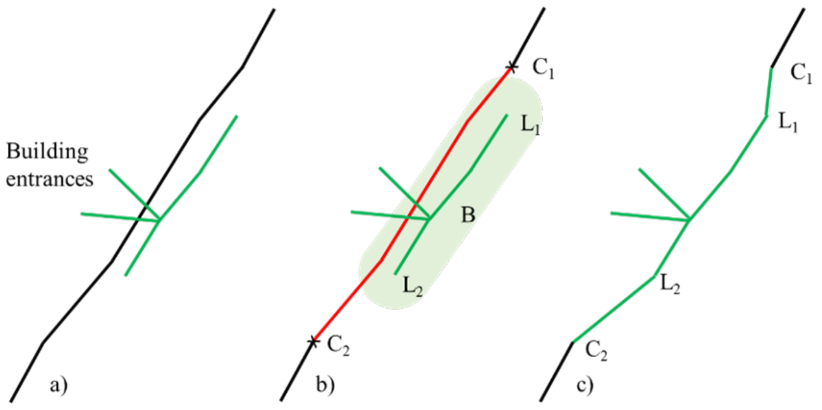


Figure 9. Integration of the LiDAR data in the model: (**a**) map obtained by means of LiDAR (green) and from the OSM (black), (**b**) generation of the buffer and detection of the connection points, (**c**) generation of connection lines between both maps.

4.3. Detection of Mobility Barriers Using Hardware Sensors on Mobile Devices

4.3.1. Mobile Application for Data Capture

In order to have access to the hardware data from mobile citizens in a smart city and convert the sensed data into knowledge about physical barriers paying and impact on human mobility, an Android application that allows the capture and processing of sensor data have been fully developed and trained. The application makes use of a Convolutional Neural Network (CNN) deep learning model trained from data samples from hardware sensors and classifies new samples in order to extract useful information about the presence of physical obstacles.

The hardware sensors used to capture the data to feed the CNN are:

- Gyroscope.
- Magnetometer.
- Accelerometer.
- Gravity Sensor.

Hardware sensors generate continuous streams of data. In order to feed the CNN, a moving windowing schema has been used. An overlap of 50% has been used in order to better detect objects which may not be aligned with the time windows.

The sensor data is preprocessed in order to reduce noise using a 4th order low pass Butterworth filter. A normalization process is then applied to achieve faster convergence in the training of the CNN by subtracting the mean value of the sensor data in the time window and dividing each sample by the standard deviation.

4.3.2. Mobile Application Modes of Operation

The mobile application is divided into 2 main modes of operation: capture mode and detection mode. Capture mode allows the capture of sensor data into a local .csv file and the transmission of the data to a central server which will be responsible for the training of a deep learning-based machine learning model. The capture mode guides each user participating in the training data to walk through a predefined series of regions containing physical barriers and records and tags the data segments. Once a valid model has been trained combining the data from several users in order to generate a model that can adapt to new users, the model is transferred to the mobile application for detection mode. In this mode, the new data captured from the sensors when the user is traversing an uncharted zone will be used as input to the model and real-time classification of the movements made by the user will be carried out in order to detect new physical barriers (which will be sent to the central system as previously described).

4.3.3. Convolutional Neural Network Model

The Convolutional Neural Network is made up of the following layers:

- 1-dimensional convolutional layer with 64 filters, filter size = 8 and relu activation function.
- MaxPooling of size 4.
- 1-dimensional convolutional layer with 128 filters, filter size = 32 and relu activation function.
- MaxPooling of size 2.
- Dense layer of 64 neurons, relu activation function.
- Dense layer of 64 neurons, relu activation function.
- 0.25 dropout.
- Dense layer of n neurons and softmax activation function (being n the number of obstacles to train and detect by the model).

The input of each sensor is fragmented into 5-second windows and a 50% overlap is used between consecutive windows. Table 1 captures the details of the Keras (https://keras.io/) model created to implement the CNN architecture.

Table 1. CNN model structure in keras.

Layer (Type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 249, 64)	1088
max_pooling1d_1 (MaxPooling1)	(None, 62, 64)	0
conv1d_2 (Conv1D)	(None, 31, 128)	262,272
max_pooling1d_2 (MaxPooling1)	(None, 15, 128)	0
flatten_1 (Flatten)	(None, 1920)	0
dense_1 (Dense)	(None, 64)	122,944
dense_2 (Dense)	(None, 64)	4160
dropout_1 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 6)	390

The model has been trained using the following parameters:

- Batch size = 32
- Number of epochs = 100
- Optimizer = adam

The model was able to find obstacles with a 0.93 accuracy on the validation data.

4.3.4. Model Training and Transfer

The training data from participating users are sent to a central server to train a CNN based model. Each class contains several samples from hardware sensor data from different users going through a particular type of obstacle. Once a large training set with several classes is collected, a notebook in python has been developed to generate CNN based machine learning models that allow classifying the information from the hardware sensors into physical barriers such as stairs and slopes. The Tensorflow library has been used to implement the CNN based machine learning model since it can be converted into a Tensorflow Lite model that can be used by the mobile application. Once the model has been generated, it would only be necessary to send it to the mobile application to use it in the detection mode.

Figure 10 shows the mobile application in detection mode showing the detected obstacles on a map.

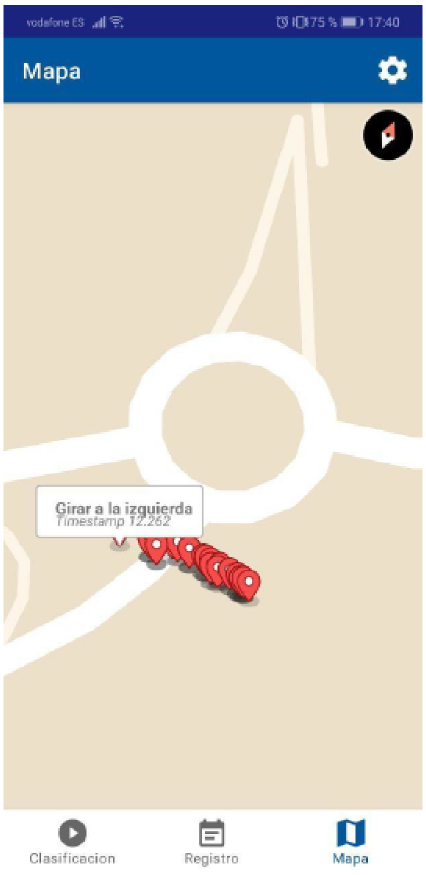


Figure 10. Obstacles detected.

4.3.5. Integration with the FLATCity System

The physical barriers detected when using the mobile application in detection mode are sent to the FLATCity central system which will use them for accessible route calculations. The CNN network will assign a probability to each data fragment estimating the likelihood that the data fragment matches each of the target barriers. If one of the estimated probabilities is above a matching threshold, a JSON object will be created to contain the detected information and sent to the FLATCity central servers using a REST API (http://flatcity.lbd.org.es/backend/api/entities/elements). The information sent captures the type of element detected, the date and time, the GPS coordinates of the detected element, and the probability assigned by the output layer in the CNN network (softmax function). An example of a particular staircase detected in Vigo (Spain) is:

```
{"activity": "Escaleras",
"date": "2020-10-22T09:39:27.153687",
"location": {"type": "Point", "coordinates": [42.232464, -8.729529]},
"probability": 0.95}
```

4.4. Detection of Mobility Barriers Using Twitter Data

Twitter users often write tweets related to something that they see during their travel. The goal of this component of the FLATCity system is to capture those tweets that are related to obstacles or mobility issues and to process them to identify the location and the type of the obstacle.

The Twitter subsystem [32] is a pipeline composed of the following stages: (1) data capture; (2) tweet preprocessing; (3) location extraction; and (4) validity detector. In the rest of this section, we describe briefly these stages. We focus on tweets that are written in Spanish because our experiments are performed in Spain. However, the Twitter subsystem could be adapted to work with other languages and locations by adapting the components that make use of specific linguistic and location information.

4.4.1. Data Capture

This is the initial stage of the Twitter subsystem. It is in charge of detecting tweets related to obstacles or mobility issues. This is done with the help of the tool T-Hoarder developed by Congosto et al. [33]. Data capture is done using queries that combine two types of terms: terms that represent urban elements (examples: "acera" (pavement), "paso de cebra" (pedestrian crossing), etc.) terms that describe the condition of an urban element (examples: "mal estado" (poor condition), "estrecho" (narrow), etc.). Each captured tweet must have at least a term of each of these two types. A similar approach has been used in the work of Wanichayapong et al. [30].

4.4.2. Tweet Preprocessing

This stage is in charge of removing tweets that are not written in Spanish or that are retweets or quotes. Also, it removes symbols like hashtags, emoticons, and URLs from the tweet.

4.4.3. Location Detector

There exist two mechanisms in the Twitter app to associate a position to a tweet: georeferencing and geopositioning. A georeference is a singular location, selected manually by the user from a set of predefined places known as Twitter Places (See https://blog.twitter.com/official/en_us/a/2010/twitter-places-more-context-for-your-tweets.html (visited on 24/08/2020).). A geoposition is composed of the GPS coordinates of the location where the tweet is written.

Unfortunately, it is well-known that most tweets come with no location information (in particular, over 98% of the tweets that we captured in the experiments performed to test this tool came with no location information). As the location of the mobility issue is a crucial part of the system, we combined the two mechanisms already mentioned (when available) with an algorithm that extracts the location information from the text of the tweet. Our algorithm is based on extracting the named entities from the text of the tweet and then performing a search to locate the street and the city mentioned in the tweet using a database with geographical information of Spain obtained from the Spanish National Statistics Institute (INE).

4.4.4. Validity Detector

Some of the tweets captured by the T-Hoarder tool do not report mobility issues, even if they contain both terms that represent urban elements and terms that represent the condition of an urban element. In the final stage, a classifier is executed to discard those tweets that do not report mobility issues. The classifier must be trained with a corpus of manually annotated tweets (in our experiments we have used a corpus of 560 tweets, of which 234 tweets were manually annotated as relevant and the others were annotated as non-relevant). For the classification task, each tweet is represented with a vector of features. Some of such features have been selected manually (number of words, the distance between urban element term and the condition term, number of verbs between the urban element term and the condition term, etc.), while others have been selected using the Wikipedia Miner semantic annotator [34].

4.4.5. Integration of the Twitter Detector in the FLATCity System

For the purpose of route planning, it is crucial to know the precise location of the mobility barrier. Therefore, only the information extracted from tweets with geoposition information attached is used to add information to a specific point in the network. The remaining tweets are used as alerts that can be used to further explore certain areas with any of the other techniques available in the FLATCity system (LiDAR and mobile sensors).

5. Case Study

5.1. Experimental Setup

To prove the viability of FlatCity, we designed a case study that includes the collection of real data and the computation of accessible routes in a city. For the case study, we selected the city of Vigo (Spain) since it is a large city (295,364 inhabitants in the municipality and 480,525 inhabitants in the metropolitan area in 2019). Besides, the primitive city is located on the terraces of *Monte do Castro*, which causes the city to abound in ramps and flights of stairs to save unevenness.

The specific case study focuses on a person who lives at *Av. das Camelias 2* (https://www.openstreetmap.org/#map=19/42.23392/-8.72876) and who must go to a bank office located at *Av. da Florida 35* (https://www.openstreetmap.org/#map=19/42.21870/-8.73563). Hence, the city blocks of *Av. das Camelias 50* and *Av. da Florida 35* are considered the target blocks to be analyzed using LiDAR and the mobile application to detect accessibility barriers.

5.2. Data Collection

We collected data from OpenStreetmap on October 19th, 2020. Figure 11 shows the area of study defined by the bounding box ((8.76857, 42.20148), (8.62275, 42.26849)). Table 2 shows the number of elements of each type imported from OpenStreetMap. We decided to import three types of destinations: banks, pharmacies, and supermarkets. The mobility-reduced parking places were not available in OpenStreetMap, hence we decided to import them ourselves using the information from the Open Data Portal of the Vigo City Council, before downloading them with our *OSM Harvester* component.

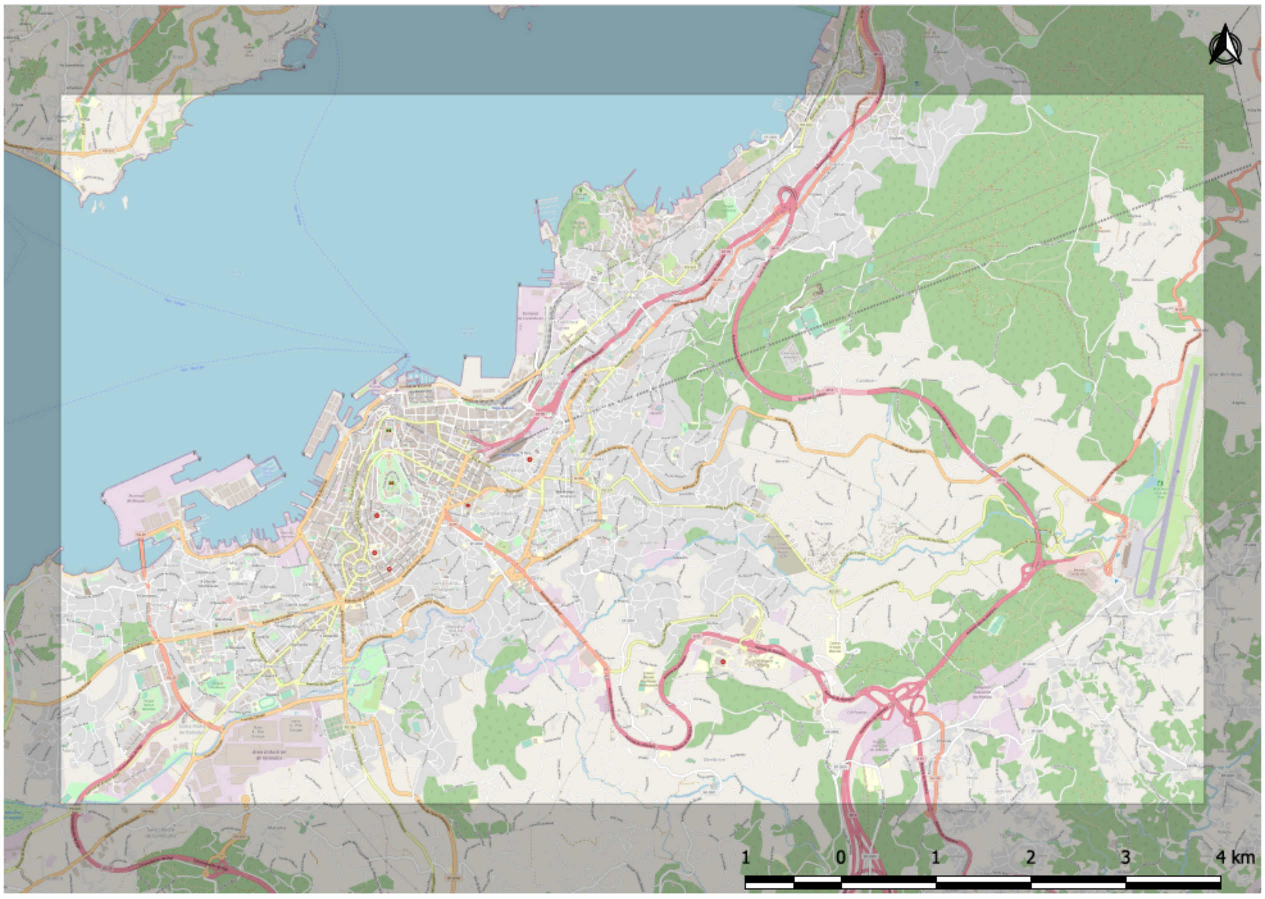


Figure 11. Area of study.

Table 2. Elements imported from OpenStreetMap.

Element Type	Count
Pedestrian crossings	1254
Road segments	4555
Pedestrian paths	1476
Steps	282
Banks	79
Pharmacies	140
Supermarkets	71
Parking places	642

The LiDAR data were acquired with a Lynx Mobile Mapper [35]. The point clouds of Av. da Florida contained 7.6 and 18 million points and point clouds of Av. das Camelias contained 5 and 9.5 million points. The average point density of the sidewalks is 2000 points per square meter with punctual variations due to the distance to the MLS and occlusions. The method of extracting information from LiDAR data abovementioned depends on two main parameters. The maximum height of steps and curbs h was limited to 25 cm based on the built environment, considering it to be an acceptable height for one person to climb. The separation between the sidewalk nodes was established in d = 5 m, which offers a compromise solution between the precision obtained from OSM and the work of Reference [31].

Table 3 shows the edges that form the graph in the walking network. The second column describes the elements in the network (i.e., walkable edges and pedestrian crossings) after executing the process described in Section 4.1. The third and the fourth columns describe the elements that were found after processing the LiDAR data and the elements collected from OpenStreetMap that have to be deleted, respectively. Finally, the fifth column describes the final amount of elements of each type in the network.

Table 3. Elements in the walking network.

Element Type	OSM Imported	LIDAR Detected	LiDAR Removed	Total
Walkable edges	14,244	226	14	14,456
Pedestrian crossings	2978	2	4	2976
Total	**17,222**	**228**	**18**	**17,432**

The mobile application to detect mobility barriers was used on the city blocks that were previously considered as the detailed area of interest. Four different segments of walking, around 100 m in length, were recorded. Eight additional elements of type stairs were detected and included in the *accessibility data model* as possible mobility barriers.

Finally, the component *T-Hoarder* was left running from 24th July 2020 to 4th November 2020. During that time, it captured 622 relevant tweets (i.e., tweets describing mobility barriers in Spain). Table 4 describes the different types of tweets that were captured. Unfortunately, only seven of the tweets captured were located in the City of Vigo, and none of them were located in the detailed area of study. This was expected because it would have been a huge coincidence that a mobility problem happened during the study period and that somebody tweeted about it. Hence, we decided to send ourselves eleven tweets in the detailed area of study to validate that they were captured and displayed.

Table 4. Tweets captured by *T-Hoarder*.

Geolocalization Type	Tweet Count
Geolocalized (exact coordinates)	7
Georeferenced (approximate coordinates)	26
Natural language processing (street found)	240
Natural language processing (without street)	349

5.3. Route Computation

To start the test, users store the location of their homes in their profile and the location where they parked their cars using an specific map-based tool. Then, they use the panel shown in Figure 12. In the panel, the user specifies the origin and the destination by clicking on the map or selecting from a suggestion list. The *home* button can be used to indicate that the route must start or stop at the home location. The first check box between the origin and the destination can be used to indicate whether the route starts from the user's car (i.e., the check box *Walk to my car* is left unchecked) or the user must first walk to the location where the car was parked. The second checkbox can be used to locate a reduced-mobility parking place close to the destination and add an additional walking segment from the parking to the destination, or whether the user will try to find a parking place upon reaching the destination. Additionally, the user can indicate that all the route will be done walking selecting the *walking* icon in the top of the panel. When the route is computed it is displayed in the map as it is shown in Figure 13. The walking segments are shown in blue and the driving segments in green.

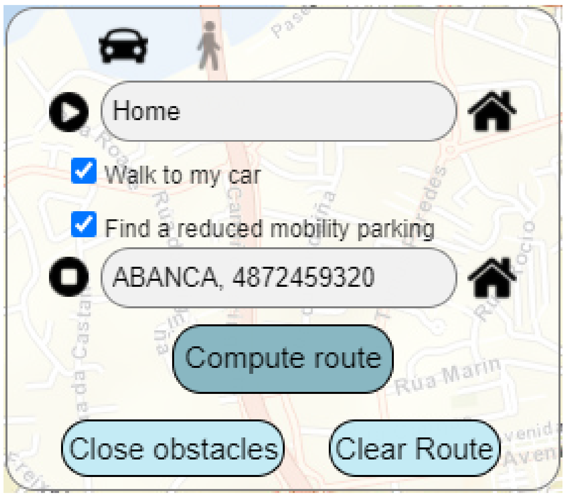


Figure 12. Configuring the route from the home of the user to a bank office at *Av. da Florida 35*.

Figure 14 shows further details of the route shown in Figure 13. Figure 14a shows the start of the route with the user walking from home to the location where the car was parked (in blue) using pedestrians crossings to change the side of the road. Figure 14b shows the end of the route. The walking segment of the route is too long because the pedestrian crossing close to the reduced-mobility parking where the car should be parked is missing in the OpenStreetMap data (as many others).

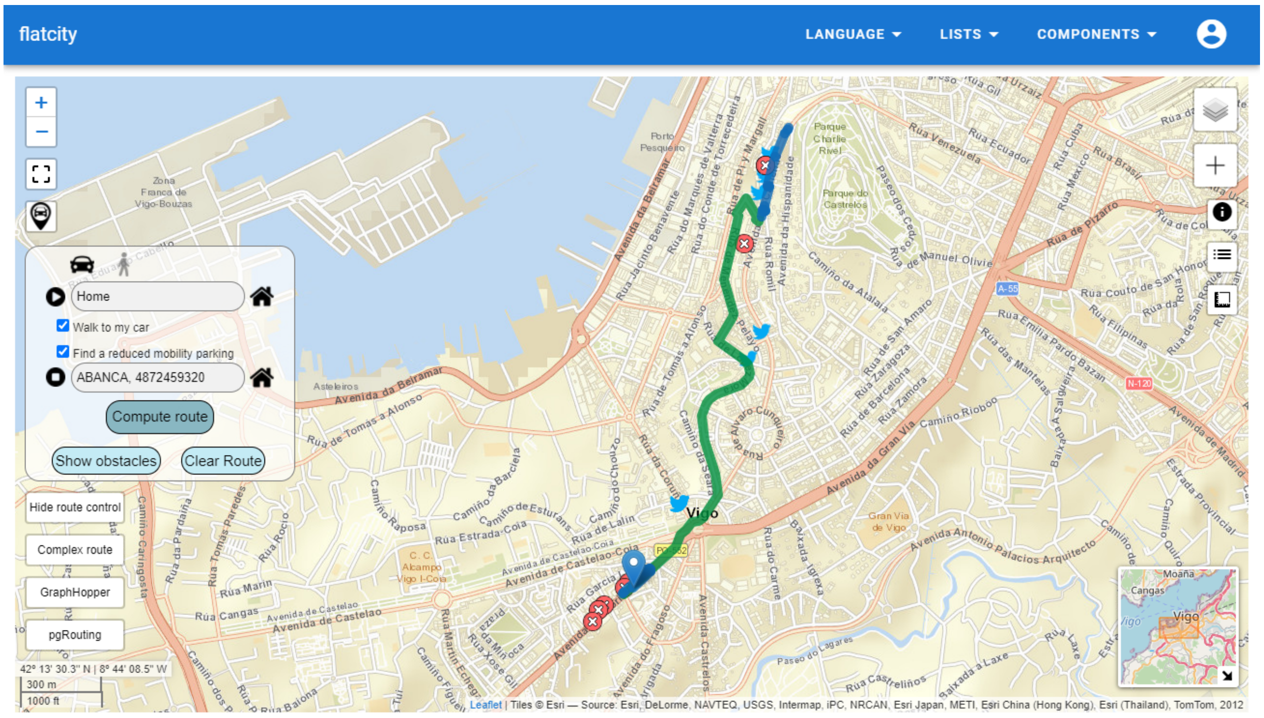
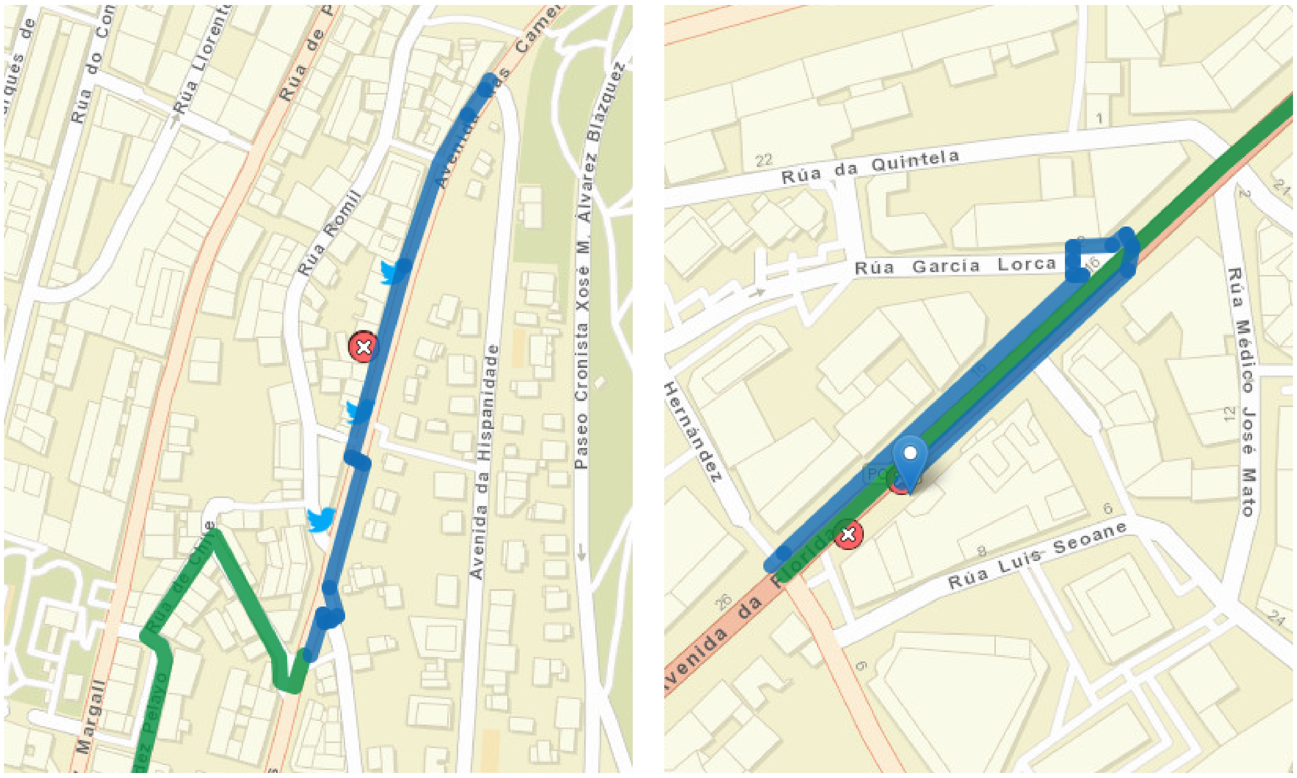


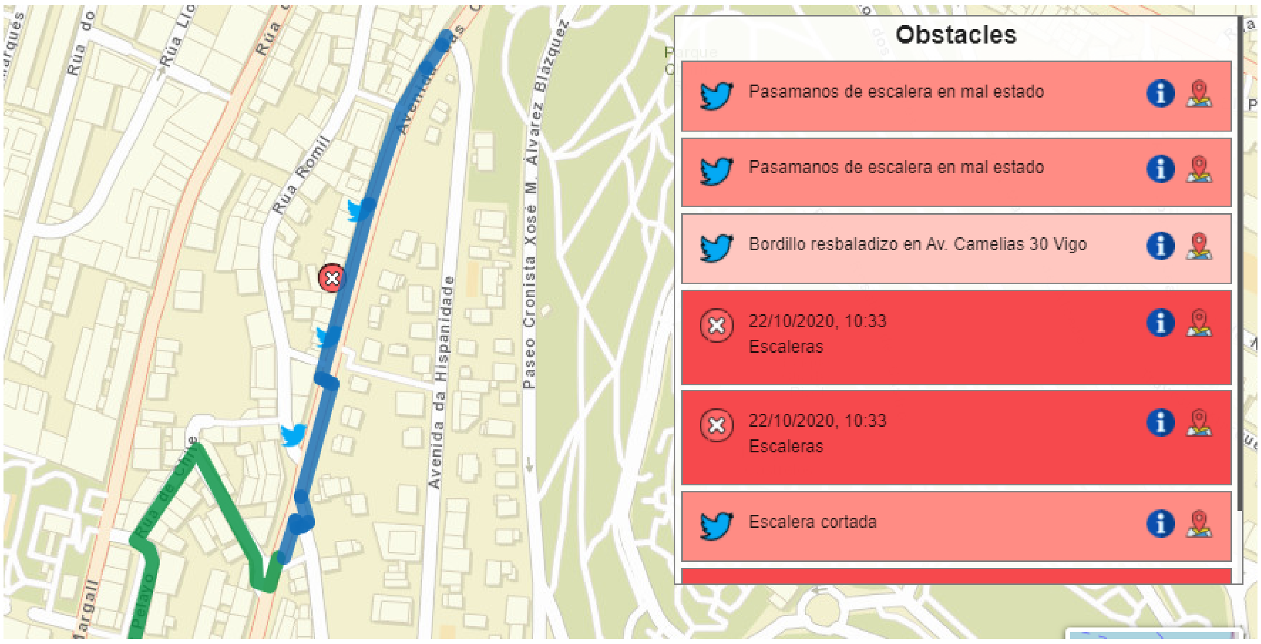
Figure 13. Complete route from the home of the user to a bank office at *Av. da Florida 35*.



(**a**) Start of the route. (**b**) End of the route.

Figure 14. Details of the route shown in Figure 13.

Figure 15 shows the obstacles that were detected in the first part of the route to *Av. da Florida 35*. The list of obstacles that can affect the route is shown on the left side of the map using color-coding to indicate the reliability of the obstacle. It shows three of the tweets that were created by and two obstacles detected by the mobile application. Further details of the obstacle can be shown (e.g., the original Tweet).


Obstacles

Pasamanos de escalera en mal estado

Pasamanos de escalera en mal estado

Bordillo resbaladizo en Av. Camelias 30 Vigo

22/10/2020, 10:33
Escaleras

22/10/2020, 10:33
Escaleras

Escalera cortada

Figure 15. Obstacles at the first part of the route to *Av. da Florida 35*.

5.4. Discussion

The case study that we have designed and executed has allowed us to test the viability of FlatCity. We have verified that OpenStreetMap is not designed to perform route calculations for people walking since there is no data related to the sidewalk network. Furthermore, OpenStreetMap recommends to introduce pedestrian crossings as points in the road network without connecting sidewalk sections, hence pedestrian crossings in OpenStreetMap cannot be used for routing. Finally, although the amount of information available is significant, it is far from perfect. In our case, we had to add to OpenStreetMap the disabled parking spaces provided by the Vigo City Council in its open data portal (https://datos-ckan.vigo.org/dataset/parking-discapacitados). As a positive aspect of OpenStreetMap, we have to emphasize that its open nature allows to propose new features and improve the quality of the data easily.

Scanning city sections with a mobile LiDAR scanner has proven very valuable. It is an effective method for improving the sidewalk network, as well as detecting pedestrian crossings and obstacles on the sidewalk. In addition, point clouds allow the recognition of other elements of the road network [36,37]. However, the cost of the equipment and the effort to apply it to a whole city is still high, and hence it could be used initially in selected areas with greater interest (e.g., because of the higher flow of people).

Retrieving obstacles using user-volunteered information and a mobile application has proven very valuable to detect physical barriers. Similarly, detecting mobility barriers using Twitter Data has also proven valuable even though the number of geolocalized and georeferenced tweets is very low compared to the number of tweets captured (less than 10%). The reason is that Twitter has decided that it does not have any interest on precisely located tweets and it is not possible to create such tweets anymore and a different client application has to be used. However, one can foresee that integrating a Twitter client that produces geolocalized tweets in the mobile application of a Smart City could serve two purposes: on the one hand users would report obstacles more frequently, and on the other hand the tweets created would be useful for FlatCity and other applications.

We have tried to compare the results of our system with existing applications. We did not find any application that could compute accesible routes in a city. Wheelmap.org (https://wheelmap.org/) is a map for finding wheelchair accessible places by marking public places with their accessibility level (i.e., fully, partly and not accessible). It also has a mobile application to browse and use the information. However, it does not record information about the sidewalks and it does not provide route finding functionality nor information regarding obstacles in the street. Google Maps (http://maps.google.com/) records information regarding the accessibility of public transit stations and routes, as well as public places. However, the pedestrian route computation uses the road network and it does not have information regarding sidewalks.

6. Conclusions

Different infrastructure elements and obstacles in the streets affect the mobility of people in smart cities in particular ways. This paper presents a novel architecture to build an information system that collects information from many heterogeneous data sources, integrates the information into a model of the city infrastructure, and compute routes adapted to each person that wants to move inside a smart city, either walking or by car. The architecture is based on a multi-sensor approach for holistic data capture, easy to use user interfaces, and optimized data storage and processing. The proposed architecture leverages available open data (from sources such as OpenStreetMap) with crowd-sensed information (from mobile apps and Twitter) and it uses additional ad-hoc data sources (based on LiDAR) to create city mobility maps from which user adapted routes can be computed. These mobility maps include information not only about different obstacles detected but also capture an estimation about the severity in which they affect the mobility of different people influenced by mobility limitations. Some of the data sources are of an asynchronous nature (i.e., OpenStreetMap and LiDAR) while other are synchronous (i.e., the user-volunteered sensor data and the issues reported with the mobile application, and the Tweets extracted from the Twitter Streaming API), thus building an accessibility data model that captures both the static and the dynamic aspects of urban mobility. The manuscript has also presented a case study in which the proposed architecture has been validated showing promising results for the future conception of personalized route recommendations.

The architecture and the the processes of the information system, that were described in the paper, contribute to the fields of spatial data mining (e.g., building a network of pedestrian infrastructures from OpenStreetMap information), large scale geospatial processing (e.g., extracting ramps, steps, and pedestrian crossings from mobile-sensed LiDAR data), geospatial data fusion (e.g., integrating the information collected from OpenStreetMap with the information obtained from the LiDAR data), and geospatial NLP (e.g., detecting accessibility problems with software sensors in social networks such as Twitter).

This work opens new and interesting lines of future work. On the one hand, additional open data sources beyond OpenStreetMap have to be considered an included in the conflation process (e.g., the information that wheelmap.org (https://wheelmap.org/) stores in OpenStreetMap regarding details on the indoor accessibility of the destinations, or the open data information provided by cities). On the other hand, additional information could be obtained from the LiDAR point clouds or the crowd-sensed information (e.g., slopes of the ramps, material of the pavement). Similarly, additional social networks could be sensed to obtain more information, even though the current trend of their owners is to limit the open-access to the information. Furthermore, additional synchronous data sources must be considered to capture more precisely the real-time or near real-time aspects of mobility (e.g., a work in progress a sidewalk that is undergoing a repair). As an example, the repair crew of the city could use the mobile application described in Section 4.3 to provide additional information regarding obstacles and accessibility issues. Finally, the possibility of using the information from one source (e.g., Twitter) to adjust the sensibility from another source (e.g., OpenStreetMap) must also be explored.

Author Contributions: All authors have contributed equally. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the project *Friendly barrierLess AdapTable City (FLATCity)* (Ministerio de Ciencia, innovación y Universidades/ERDF, EU) funded by the Spanish Agencia Estatal de Investigación (AEI, doi 10.13039/501100011033), and in part by the European Regional Development Fund (ERDF), under Grants TIN2016-77158-C4-1-R, TIN2016-77158-C4-2-R and TIN2016-77158-C4-3-R. This work was also supported in part by the project *Massive Geospatial Data Storage and Processing for Intelligent and Sustainable Urban Transportation (MaGIST)*, funded by the Spanish Agencia Estatal de Investigación (AEI, doi 10.13039/501100011033) under grants PID2019-105221RB-C41, PID2019-105221RB-C43 and PID2019-105221RB-C44. The research of Miguel R. Luaces was also partially founded by: Xunta de Galicia/FEDER-UE GRC: ED431C 2017/58 and Xunta de Galicia/FEDER-UE, ConectaPeme, GEMA: IN852A 2018/14. Miguel R. Luaces also wishes to acknowledge the support received from the Centro de Investigación de Galicia "CITIC", funded by Xunta de Galicia and the European Union (European Regional Development Fund- Galicia 2014-2020 Program), by grant ED431G 2019/01. Jesús Balado and Lucía Díaz-Vilariño would like to thank to the Xunta de Galicia given through human resources grants ED481B-2019-061 and ED481D 2019/020, respectively.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Panta, Y.R.; Azam, S.; Shanmugam, B.; Yeo, K.C.; Jonkman, M.; De Boer, F.; Alazab, M. Improving Accessibility for Mobility Impaired People in Smart City using Crowdsourcing. In Proceedings of the 2019 Cybersecurity and Cyberforensics Conference (CCC), Melbourne, Australia, 8–9 May 2019; pp. 47–55.
2. Saha, M.; Saugstad, M.; Maddali, H.T.; Zeng, A.; Holland, R.; Bower, S.; Dash, A.; Chen, S.; Li, A.; Hara, K.; et al. Project sidewalk: A web-based crowdsourcing tool for collecting sidewalk accessibility data at scale. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, Glasgow, UK, 4–9 May 2019; pp. 1–14.
3. Ding, C.; Wald, M.; Wills, G. A survey of open accessibility data. In Proceedings of the 11th Web for All Conference, Seoul, Korea, 7–11 April 2014; pp. 1–4.
4. Suleymanov, T.; Kunze, L.; Newman, P. Online Inference and Detection of Curbs in Partially Occluded Scenes with Sparse LIDAR. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; pp. 2693–2700.
5. Serna, A.; Marcotegui, B. Urban accessibility diagnosis from mobile laser scanning data. *ISPRS J. Photogramm. Remote. Sens.* **2013**, *84*, 23–32, [CrossRef]
6. Balado, J.; Díaz-Vilariño, L.; Arias, P.; González-Jorge, H. Automatic classification of urban ground elements from mobile laser scanning data. *Autom. Constr.* **2018**, *86*, 226–239, [CrossRef]
7. Yang, F.; Liang, Y.; Li, D.; Su, F.; Zhu, H.; Zuo, X.; Li, L. *Detection of Space Connectivity from Point Cloud for Stair Reconstruction*; Technical Report; EasyChair: Manchester, UK, 2019.
8. Balado, J.; van Oosterom, P.; Díaz-Vilariño, L.; Meijers, M. Mathematical morphology directly applied to point cloud data. *ISPRS J. Photogramm. Remote. Sens.* **2020**, *168*, 208–220. [CrossRef]
9. Hou, Q.; Ai, C. A network-level sidewalk inventory method using mobile LiDAR and deep learning. *Transp. Res. Part C Emerg. Technol.* **2020**, *119*, 102772. [CrossRef]
10. Ai, C.; Tsai, Y. Automated sidewalk assessment method for americans with disabilities act compliance using three-dimensional mobile lidar. *Transp. Res. Rec.* **2016**, *2542*, 25–32. [CrossRef]
11. Balado, J.; Díaz-Vilariño, L.; Arias, P.; Lorenzo, H. Point clouds for direct pedestrian pathfinding in urban environments. *ISPRS J. Photogramm. Remote. Sens.* **2019**, *148*, 184–196. [CrossRef]
12. Schmittwilken, J.; Plümer, L. Model-based reconstruction and classification of facade parts in 3D point clouds. *Int. Arch. Photogramm. Remote. Sens. Spat. Inf. Sci.* **2010**, *38*, 269–274.
13. Balado, J.; Díaz-Vilariño, L.; Arias, P.; Garrido, I. Point clouds to indoor/outdoor accessibility diagnosis. *ISPRS Ann. Photogramm. Remote. Sens. Spat. Inf. Sci.* **2017**, *4*, 287–293. [CrossRef]

14. Soilán, M.; Riveiro, B.; Sánchez-Rodríguez, A.; Arias, P. Safety assessment on pedestrian crossing environments using MLS data. *Accid. Anal. Prev.* **2018**, *111*, 328–337. [CrossRef]
15. Soilán, M.; Riveiro, B.; Martínez-Sánchez, J.; Arias, P. Segmentation and classification of road markings using MLS data. *ISPRS J. Photogramm. Remote. Sens.* **2017**, *123*, 94–103. [CrossRef]
16. Díaz Vilariño, L.; Boguslawski, P.; Khoshelham, K.; Lorenzo, H.; Mahdjoubi, L. Indoor navigation from point clouds: 3d modelling and obstacle detection. *ISPRS Int. Arch. Photogramm. Remote. Sens. Spat. Inf. Sci.* **2016**, *XLI-B4*, 275–281. [CrossRef]
17. Maruyama, T.; Kanai, S.; Tada, M. Simulation-Based Evaluation of Ease of Wayfinding Using Digital Human and As-Is Environment Models. *ISPRS Int. J. Geo-Inf.* **2017**, *6*, 267. [CrossRef]
18. Oßwald, S.; Gutmann, J.S.; Hornung, A.; Bennewitz, M. From 3D point clouds to climbing stairs: A comparison of plane segmentation approaches for humanoids. In Proceedings of the 2011 11th IEEE-RAS International Conference on Humanoid Robots, Bled, Slovenia, 26–28 October 2011; pp. 93–98.
19. Luo, R.C.; Hsiao, M.; Liu, C.W. Multisensor integrated stair recognition and parameters measurement system for dynamic stair climbing robots. In Proceedings of the 2013 IEEE International Conference on Automation Science and Engineering (CASE), Madison, WI, USA, 17–20 August 2013; pp. 318–323.
20. Capponi, A.; Fiandrino, C.; Kantarci, B.; Foschini, L.; Kliazovich, D.; Bouvry, P. A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2419–2465. [CrossRef]
21. Casilari, E.; Álvarez-Marco, M.; García-Lagos, F. A Study of the Use of Gyroscope Measurements in Wearable Fall Detection Systems. *Symmetry* **2020**, *12*, 649. [CrossRef]
22. Deb, S.; Yang, Y.O.; Chua, M.C.H.; Tian, J. Gait identification using a new time-warped similarity metric based on smartphone inertial signals. *J. Ambient. Intell. Humaniz. Comput.* **2020**, *11*, 4041–4053. [CrossRef]
23. Lara, O.D.; Labrador, M.A. A survey on human activity recognition using wearable sensors. *IEEE Commun. Surv. Tutor.* **2012**, *15*, 1192–1209. [CrossRef]
24. Brodie, M.A.; Coppens, M.J.; Lord, S.R.; Lovell, N.H.; Gschwind, Y.J.; Redmond, S.J.; Del Rosario, M.B.; Wang, K.; Sturnieks, D.L.; Persiani, M.; et al. Wearable pendant device monitoring using new wavelet-based methods shows daily life and laboratory gaits are different. *Med Biol. Eng. Comput.* **2016**, *54*, 663–674. [CrossRef]
25. Hu, S.; Su, L.; Liu, H.; Wang, H.; Abdelzaher, T.F. Smartroad: Smartphone-based crowd sensing for traffic regulator detection and identification. *ACM Trans. Sens. Networks (TOSN)* **2015**, *11*, 1–27. [CrossRef]
26. Sakaki, T.; Okazaki, M.; Matsuo, Y. Earthquake shakes Twitter users: Real-time event detection by social sensors. In Proceedings of the 19th International Conference on World Wide Web, Raleigh North, CA, USA, 26–30 April 2010; pp. 851–860.
27. Congosto, M.; Fuentes-Lorenzo, D.; Sánchez-Fernández, L. Microbloggers as Sensors for Public Transport Breakdowns. *IEEE Internet Comput.* **2015**, *19*, 18–25. [CrossRef]
28. Anastasi, G.; Antonelli, M.; Bechini, A.; Brienza, S.; D'Andrea, E.; De Guglielmo, D.; Ducange, P.; Lazzerini, B.; Marcelloni, F.; Segatori, A. Urban and social sensing for sustainable mobility in smart cities. In Proceedings of the 2013 Sustainable Internet and ICT for Sustainability (SustainIT), Palermo, Italy, 30–31 October 2013; pp. 1–4.
29. Kumar, A.; Jiang, M.; Fang, Y. Where not to go? Detecting road hazards using Twitter. In Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, Gold Coast, Australia, 6–11 July 2014; pp. 1223–1226.
30. Wanichayapong, N.; Pruthipunyaskul, W.; Pattara-Atikom, W.; Chaovalit, P. Social-based traffic information extraction and classification. In Proceedings of the 2011 11th International Conference on ITS Telecommunications, St. Petersburg, Russia, 23–25 August 2011; pp. 107–112.
31. López Pazos, G.; Balado Frías, J.; Díaz Vilariño, L.; Arias Sánchez, P.; Scaioni, M. Pedestrian pathfinding in urban environments: Preliminar results. In Proceedings of the Geospace 2017, Kyiv, Ucrania, 4–6 December 2017.
32. Sánchez-Ávila, M.; Mouriño-García, M.A.; Fisteus, J.A.; Sánchez-Fernández, L. Detection of Barriers to Mobility in the Smart City Using Twitter. *IEEE Access* **2020**, *8*, 168429–168438. [CrossRef]
33. Congosto, M.; Basanta-Val, P.; Sanchez-Fernandez, L. T-Hoarder: A framework to process Twitter data streams. *J. Netw. Comput. Appl.* **2017**, *83*, 28–39. [CrossRef]
34. Milne, D.; Witten, I.H. An open-source toolkit for mining Wikipedia. *Artif. Intell.* **2013**, *194*, 222–239. [CrossRef]
35. Puente, I.; González-Jorge, H.; Martínez-Sánchez, J.; Arias, P. Review of mobile mapping and surveying technologies. *Measurement* **2013**, *46*, 2127–2145. [CrossRef]
36. Holgado-Barco, A.; Riveiro, B.; González-Aguilera, D.; Arias, P. Automatic inventory of road cross-sections from mobile laser scanning system. *Comput. Aided Civ. Infrastruct. Eng.* **2017**, *32*, 3–17. [CrossRef]
37. Balado, J.; González, E.; Arias, P.; Castro, D. Novel approach to automatic traffic sign inventory based on mobile mapping system data and deep learning. *Remote Sens.* **2020**, *12*, 442. [CrossRef]