Digital Twin Framework for Smart City Solutions

L. Adreani[1], P. Bellini[1], C. Colombo[2], M. Fanfani[1,2], P. Nesi[1], G. Pantaleo[1], R. Pisanu[2]
University of Florence, Florence, Italy, email: <name>.<surname>@unifi.it
1) DISIT lab, https://www.disit.org, https://www.snap4city.org
2) Computational Vision Group http://cvg.dsi.unifi.it/cvg/

***Abstract*—Recently, 3D city modelling has attracted a growing interest as a building block for creating city Digital Twins. They are complex representations that include interactive representations of buildings and infrastructures, integrated with the wide range of data typically useful in a Smart City environment. This paper presents an automatic method for producing 3D city models from a various set of data, as well as its integration into the open-source Smart City framework, Snap4City. The proposed solution offers a method for creating effective integrated data visualizations of 3D city entities coupled with a large variety of Smart City data (e.g., IoT Devices which generate time-series data, heatmaps, geometries and shapes related to traffic flows, bus routes, cycling paths). The solution is based on a deep learning approach for rooftop detection and alignment based on a U-Net architecture. The implementation has been enforced into the open-source Snap4City Smart City platform, and has been validated by using a manually created ground-truth of 200 buildings scattered uniformly in the central area of Florence, plus a number of meshes representing a number of facades (not detailed in this paper), and traffic flows, pins, heatmaps, etc.**

Keywords—3D City model, Photorealistic texture, digital twin, Smart City applications.

I. INTRODUCTION

Smart Cities are complex infrastructures integrating multiple linked data sources, Internet of Things (IoT) devices and applications, involving many different data and stakeholders. In this context, spatial data information may act as enabler for smart applications and decision support systems, provided that they are interoperable with legacy and future solutions [1]. Recently, the aspects related to digital 3D city modelling and digital twin have gained a growing interest, since they allow to create a more realistic context in which the decision makers can perform analyses, simulations, planning and monitoring in several different domains and application areas (e.g., urban planning, energy management, traffic and mobility, disaster management, air pollution monitoring). Many approaches have been proposed in literature, such as: CityGML, CityJSON, the combination of Building Information Modeling (BIM) and Geographic Information System (GIS) providing a City Information Modeling (CIM) [2].

In the past years, a relevant amount of research has been made in the field of 3D city modelling, to recreate realistic visualizations. However, due to the typical size of a city, handling all the data and their processing is a challenging task still remained unsolved [4]. One critical aspect of developing a high-fidelity 3D city model is to find the correct model and format for the data that can be rendered by a visual interface and may be on web browser. For this purpose, a set of requirements have been proposed by CityGML, according to different levels of detail (LoD) which can be addressed by the models. According to [3], there are five levels of detail: LoD0 is represented by those models having only a 2D map with 3D terrain; LoD1 presents buildings as simple boxes; LoD2 adds rooftops details to LoD1 buildings; LoD3 presents also external facades structure; LoD4 adds building interiors. LoD4 was introduced in CityGML 2.0, but it was removed in the latest version of CityGML 3.0. The CityGML and CityJSON have defined a format for the representation of geometry and topology for 3D buildings, using respectively XML and JSON. CityGML 3.0 integrates a BIM standard, alongside the GIS (Geographical Information Systems) format, from Industrial Foundation Class (IFC) [5]. Some integrations of CityGML have been proposed in real cases, such as the city of Helsinki, in which a LoD3 city model was implemented and made publicly available [6]. However, the system do not provide integration with IoT data or other kind of city data. Another similar integration was made by the city of Rotterdam [27], recreating a LoD2 type of buildings, however neither integrating any decoration elements nor elevation of terrain (this is relevant aspect for non flat cities). An attempt of making a LoD3 3D city model was made by ETH Zurich with the VarCity [7]. However, the provided semantic information is generally limited to a small number of semantic classes. The **3dcitydb** implements a 3D model for the city of Berlin [28], providing a pickable model of LoD2 buildings, supporting also WMS (Web Map Service) layers (typical of GIS solutions providing maps, heatmaps and orthomaps) and terrain layer. However, neither of those are provided. The city of Stockholm [29] implements many aspects of Digital Twin concept, such as POI (point of interest), LoD3 type buildings, either with 3D tiles and a modelled one, and others 3D entities. However, the solution lacks in the implementation of any WMS heatmap.

In the context of 3D city data collection, advancements in Light Detection And Ranging (LiDAR) technology allow to model urban topography at spatial resolution and granularity which were not achievable before the advent of this technology [8]. In [9], a method to create a city model from a point cloud generated by LiDAR technology is presented. This approach has shown to reduce the time to generate the model, but it cannot process unsymmetrical objects and presents some geometrical error.

A more pleasant and realistic 3D city representations can be obtained by enhancing them with textures extracted from RGB images. In particular, rooftops textures can be obtained from orthomaps or satellite images, facades image patterns, etc. However, this is not an easy task: at first, rooftops have to be detected in the RGB images [10]; then, the segmented patches must be carefully aligned with the top-view of the 3D map.

DOI reference number: 10.18293/DMSVIVA22-012

Indeed, even if geolocalization information is typically available, errors are present due to uncertainties [11] and an accurate multi-modal registration is required (e.g., between the RGB images and the 3D structure) [12]. In the literature, several works have addressed these topics using both computer vision standard and learning-based solutions. In [13], handcrafted features and a hierarchical segmentation approach have been used to identify the buildings in rural areas. SVMs (support Vector Machines) [14] and Random Forests [15] have also been used to address this task. For example, in [16] the authors proposed a three-steps method based on color-based clustering, roof detection using an SVM and a final false negative recovery. Slightly different, in [17] a pair-wise exploitation of satellite images has been used to reconstruct a 3D model that could then be employed to identify rooftop regions. However, such solutions not only have some limitations when working on areas with dense buildings, but also require a successive registration on the 3D map. More recently, deep learning based solutions appeared for remote sensed image processing [18], [19]. In [20], a Mask R-CNN (region based convolutional neural network) [21] was used to detect rooftops from aerial images. Differently, in [22], [23] a U-Net architecture [24] has been preferred. Moreover, these last two solutions provide not only rooftop segmentation but also the registration on 3D data.

In this paper, a 3D City Modelling Framework for Smart City Digital Twin with textures is presented. The main contributions of this paper are the following: first, the production of a full functional solution showing 3D city representations on the basis of roads, building planimetry, high, detailed buildings with meshes. Thus the creation of a full automatized algorithm to map the buildings in the area, creating building models from building type and the integration of terrain aspects/pattern, more sophisticated 3D shapes based on meshes. Thus the integration of the 3D city representations into a Smart City framework (the open-source Snap4City platform), in order to provide a smart environment and applications for visualizing city entities and related data (coming, for instance, from IoT devices generating time-series data, heatmaps, geometries and shapes related to traffic flows, bus routes, cycling paths etc.), with the possibility to pick single city elements or buildings on 3D city representation, and inspect their data and attributes. The proposed solution is an open-source web-based tool for producing a global digital twin integrating IoT and many other kind of Smart City data, which has been designed to satisfy the most of the identified requirements as reported in the paper.

The paper is organized as follows: in Section II, requirements are presented. In section III, the architecture is described putting in evidence the data flow. The model is detailed in Section IV. In section V, details of the image processing solution based on machine learning is presented for producing the roofs' patterns from orthomaps. In Section I, the final result and process is presented. Some notes above the distribution of 3D information presented is described in Section VI. In Section VII, some notes on the validation regarding the process for roof patter estimation are reported. Finally, conclusion are drawn in section VIII.

II. Requirements Analysis

With the aim of creating a Digital Twin in the context of smart cities, the 3D representation of buildings in the city covers a relevant role. To this end, a set of specific requirements have been identified and are reported in this section. In the past a similar approach has been proposed by CityGML which defined different levels of detail (LoD) for the models [3]. The CityGML approach was mainly on the visual represented and it is actually not enough detailed to describe the needs of full Digital Twin models in the Smart City solutions for decision makers. *Therefore, a more complete set of requirements and an assessment model for Web delivering of 3D representations of Digital Twins at the support of a decision support system is presented in this section.* Most of the requirements are related to the 3D representation and to the integration of 3D data with the massive data infrastructure in back which actually supports the decision makers. For example, to move a bus stop, to close an area for a market, to see the impact of some event. In particular, the solution has to provide support for representing in the 3D context, the:

R1. buildings of the city as city structure, roads, gardens, etc. The single building should be represented with realistic details in terms of shape (facades, roof, towers, cupolas, etc.), and patterns on facades and roofs. To this end, different techniques can be adopted to model the buildings. For example, (i) the simple bounding box of the buildings obtained from the perimeter extruded up to the heights of the eaves, (ii) the creation of meshes precisely describing every tiny detail of the physical structure.

R2. ground information as road shapes and names, names of squares and localities, etc., exploiting the so called Orthomaps, with eventual real aerial view patterns. They are typically provided in terms of multi resolution tiled images from GIS systems using WMS protocol;

R3. one or more heatmaps superimposed (and transparent) on the ground level information without overlapping the buildings. For example, to represent some information, such as: the heatmaps of temperature, traffic flow, pollutant, people flow, etc. Also in this case, they are typically provided in term of multi resolution geolocated tiled images, provided by GIS using WMS protocol;

R4. paths and areas super-imposed on the ground and on heatmaps levels without overlapping the buildings, for example those needed to describe the perimeters of gardens, the cycling paths, the trajectories, border of gov areas, etc. This information is quite specific and has to be produced on the basis of the information recovered from some Open Data. Once recovered it can be distributed by using GIS in WFS/WMS protocols;

R5. pin marking the position of services, IoT Devices, Point of Interest, POI, Key Performance Indicator, KPI, etc., and providing clickable information according to some data model which may provide access to Time Series, shapes, etc. This information is quite specific and can be produced on the basis of the information recovered from Private and/or Open Data;

R6. terrain information and elevation, so that the skyline of the city may include the shape of eventual mountains around, and under the city as well. This also means that the

buildings and Orthomaps should be placed according to the terrain elevation.

R7. additional 3D entities for completing the realism of the scenario, such as: trees, benches, fountains, semaphores, digital signages, and any other city furniture, etc.

	CityGML [3]	Helsinki [6]	Rotterdam [27]	Berlin [28]	Stockholm [29]
R1.i	Yes (LoD1)	No (only available in higher detail)	No (only available in higher detail)	No (only available in higher detail)	No (only available in higher detail)
R1.ii	Yes (LoD3)	Yes (either with object or 3D tiles)	Yes (LoD2)	Yes (LoD2)	Yes (LoD3)
R2	No	Yes (C)	Yes (C)	Yes (C)	Yes (but with a fixed Orthomap)
R3	No	No	No	Yes (does not include Wms)	No
R4	No	Yes (C)	Yes (C)	No (x)	Yes
R5	No	No	No	No	Yes
R6	Yes	Yes (with 3D tiles)	No	No	Yes
R7	Yes (LoD2)	Yes (with 3D tiles)	No	No	Yes (3d tiles and single entity)
RA	No(*)	Yes	Yes	Yes	Yes
RB	No(*)	No	No	No	No
RC	No(*)	No	No	No	Yes
RD.1	not clear (may be)	Yes (when models are loaded as object, not if loaded as 3D tiles)	Yes	Yes	No
RD.2	No	No	No	No	No
RE	No(*)	No	No	No	Yes
RF	No(*)	Yes (**)	Yes (**)	No (x)	No

Table 1 -- Comparison of 3D representation platforms for Digital Twins vs Smart City. Where: (*) defines only the building model, (**) functionality implemented in CESIUM but without any model placed underground, (x) use CESIUM, it could be possible to integrate, (C) based on CESIUM.

In addition, the solution has to be capable to provide some interactivity on the above mentioned 3D data structures, in particular it should be capable to depict the 3D scene:

RA. according to the point of view, providing capabilities for changing it by: zoom, rotate, tilt, and pan the scene and also changing the light or time of the day/night (this may lead to produce shades), etc.

RB. with the sky, maybe with different sky conditions according to the actual day, light condition, weather, or weather forecast.

RC. providing access to the information associated with augmenting PINs: POI, KPI, etc., and maybe to real time data, and time series associated with eventual IoT Devices located on the 3D scene.

RD. providing the possibility of selecting each single building to: (1) pass at a more detailed information associated with the building, or (2) go into a BIM view of the building, with the possibility of navigating into the building structure, and again to access at the internal data associate to PINs into the building. May be also disabling the building view to see only the 3D of city without the buildings but with PINs.

RE. providing possibility of selecting an element (3D, PIN, ground, heatmap) to provoke a call back into a business logic tool for provoking events and actions in the systems, at which the developers may associate intelligence activities, analytics, other views, etc..

RF. providing the possibility of inspecting the ground terrain and see the detailed 3D elements placed in the underground, such as water pipes, or located in the ground as benches, luminaries, red lights, etc.

According to the identified requirements, in the following **Table 1,** an assessment of the most relevant solutions is reported.

III. Architecture and Process

According to the above described requirements, a solution for Smart City Digital Twin, **SCDT**, has to address three main aspects: **(a)** the **3D model** enabling the representation of the information in integrated manner, **(b)** the **software architecture for distributing** and provide access to the 3D representation via a suitable user interface presenting the (a) 3D model including the interactivities features, and **(c)** the **production process** of the 3D models by starting from multiple information which have to be recovered from accessible resources or produced/acquired,

The above described requirements from R1 to R7 mainly impact on (a) and (b) for the resulting performance on distribute and reproduce the representation in real time on browser. Thus, providing support for the users to interact with the 3D representation in real time. The system presents challenging aspects due to the large amount of data to be processed on client side on the basis of the point of view. This impacts especially when several details are provided at the same time in the same view, e.g., photorealistic textures, detailed heatmaps, complex terrains shape that implies to compute several projections to avoid overlaps, etc. In these cases, the issue is typically mitigated at the expense of a lower resolution of textures.

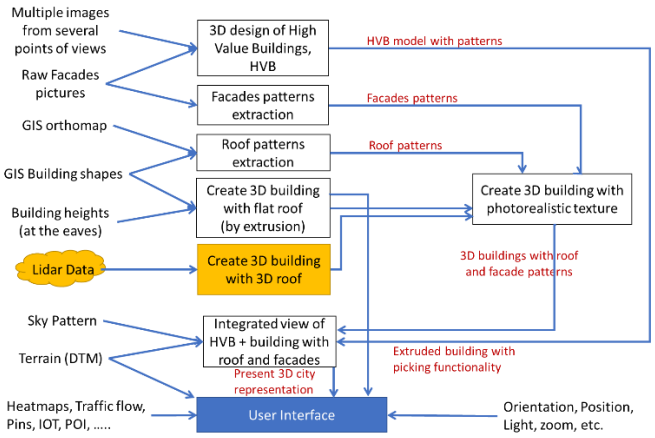

Figure 1 – Data Flow of the production process for creating a Digital Twin for smart cities.

On the other hand, the features from RA to RF have to be mainly satisfied by the production process © of the data model to be distributed according to (a) and (b). In fact, the model can be composed by several elements: 3D representation, meshes, patterns, etc. The process to pass from images and data to the integrated 3D model is not trivial as partially described in this

paper for some aspects. On this regard, the production process to produce the **SCDT** model is depicted in **Figure** 1. The production process puts in evidence the data sources: GIS, raw images, building shapes, heatmaps, PINs, POI, IoT devices, Terrain DTM (Digital Terrain Model), etc., and the optional LIDAR data which may be exploited for adding details and shortcutting some of the procedures.

According to **Figure 1**, the production process for the creation of the 3D model requires a set of sub-processes:

- **Roof pattern extraction**: photorealistic textures of building rooftops can be obtained from orthomaps. Since orthomaps are typically roughly geo-localized, a careful registration w.r.t. the building shapes is required. After that, textures can be extracted and provided as PNG or JPEG files.
- **Facades pattern extraction:** differently from rooftop textures, where the used orthomaps are relatively easily accessible nowadays, façade texturing requires a specific acquisition campaign. Moreover, the acquired RGB images must be processed to remove radial and projective distortions, and finally, the building facades must be accurately identified and extracted. As for rooftops, obtained textures can be provide as PNG or JPEG files.
- **Create 3D buildings with flat roof (by extrusion):** given the building shapes plus their height, typically measured at their eaves, simple 3D models with flat rooftop can be obtained. The resulting data format is a GeoJSON file with a height/elevation attribute to compute the building extrusion from the ground at run time. This is the model used to implement the picking functionality.
- **Create 3D building with 3D roof**: when a Digital Surface Model (DSM) is available, obtained from LIDAR data or other acquisition modality, accurate 3D roof shapes can be obtained to build a more realistic SCDT. The buildings 3D models can be provided as glTF (GL transmission format) files, with geo-localization information.
- **Create 3D building with photorealistic textures:** the 3D buildings obtained by extrusion or exploiting a DSM can be enhanced with photorealistic rooftop and façade patterns by applying textures extracted from RGB images. Textured building models are saved in glTF files, with geo-localization information.
- **3D design of High Value Buildings, HVBs**: in order to produce accurate representation of HVBs a manual 3D design or automatic computer vision techniques (such as Structure from Motion) can be employed. This requires precise measurements or specific image/video acquisition campaign. Additionally, geo-localization information must be provided. Also in this case, the resulting textured 3D models can be exported as geo-localized glTF files.
- **Integrated view of HVBs + buildings with roof and facades**: the building 3D models and the HVB models are finally placed into a unique 3D representation exploiting their geo-localization information, thus obtaining the complete 3D representation for the SCDT.

The general architecture for distributing **SCDT** includes a set of data integrating 3D models, meshes, with DTM, heatmap, traffic flow, Pins, IOT, POI, etc., as described in the paper.

For the distribution of the data:

- **3D representation File in GeoJSON via HTTPS**: it describes the 3D structure of the city and all information related to it. It is used to represent the city model in extruded mode and to retrieve the buildings information or other BIM data for the picking functionality.
- **3D representation File in glTF/GLB (GLB is the binary version of glTF) via HTTPS**: it describes the 3D structure of the city in terms of building and their relationships with the other graphic elements: facades, meshes of HVB, textures and materials.
- **Pattern files via HTTPS**: pattern images for facades, roofs, DTM files in PNG format, Sky texture, etc.,
- **GIS server via WMS** over https is providing (via GeoServer, also integrated into Snap4City platform): orthomaps, maps, heatmaps, animated heatmaps, traffic flows, animated traffic flows, etc., on the basis of the portion of the map shown in the window frame.
- **SuperService Map of Snap4City platform via smart city API** via https [33], [30] is providing semantic details in JSON such as: roads graph, POI, IoT data, Pins, cycling paths, vectorial traffic flows, etc., on the basis of the portion of the map shown in the window.

IV. Model and Representation

The model for creating the 3D representations, which allows to provide all the above mentioned information, is based on a hierarchical layered structure depicted in **Figure 2** and described in this sections.

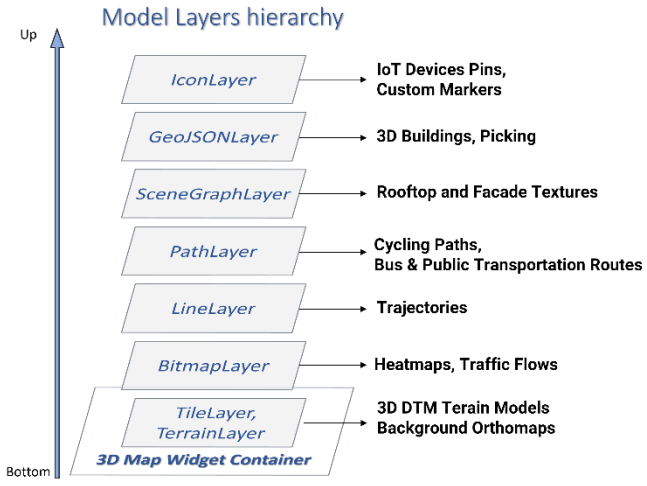

Figure 2: Hierarchical layers structure of the model

The layered solution has been implemented via WebGL API, in order to process all the data in parallel, thanks to the GPU passthrough, to this end, the open-source library called Deck.gl has been used. All the layers needed for the representation of the

Snap4City platform data types have been implemented, and they are loaded at runtime on user demand. Thanks to the multi-layer structure of deck.gl, layers have been implemented individually with their own safe context, to avoid interferences one with each other. Every layer has its own scope, managing its own data type. Therefore, in the following we are introducing the implemented layers to describe data types provided in the Snap4City 3D representation.

First, the base deck application has been realized by using a custom implementation and management of the viewState object, in which all the geographical information for the map (such as latitude, longitude, zoom, etc.), are defined. We have also implemented a custom rendering in order to add features like SkyBox that need direct access to the WebGL context. Starting from the first layer. The elevation of the terrain has been modelled by implementing a composite layer called TileLayer, which is used to divide the maps in multiple tiles with their own sublayer: for each tile a sublayer called TerrainLayer has been created. Thus the elevation map, in the form of DTM files, has been used to create the TerrainLayer 3D model from the map, and the background orthomap has been used as a texture of the terrain objects. The result is a 3D representation of terrain with texture to better represent the territory.

The background orthomaps have been also implemented through a TileLayer. In this case, we used the BitmapLayer to display an image in the map. This method has been also used to represent heatmaps, which are essentials to provide a fast access / representation to large amounts of data. In order to implement heatmap visualization in deck.gl, we used the composite layer which automatically retrieves heatmaps from a dedicated geo-server (through several formats, including WMS) and displays them as an image. Heatmaps can be static or animated; static heatmaps are viewed as single PNG images, while animated ones are sent by the geo-server in GIF format, and they are later divided in multiple images and rendered sequentially with a customizable delay time.

For the implementation of data coming from different sources like IoT devices, trajectories, cycling paths, etc., various layers with a specific JSON mapping have been implemented. To display paths and geometries, different layers depending on the type of geometry to be displayed have been used, i.e. LineLayer for trajectories, PathLayer for the cycling path. IoT devices are also displayed as pickable markers on the map. When a user selects one of them, a popup with the sensor information (static attributes as well as real-time data, if available) is shown. Whenever the sensor provides real-time data, they can be displayed on dedicated widgets, such as time trends, when the user requests them.

3D representation of buildings are provided in two manners: Extruded and Realistic (meshes, HVB). Extruded buildings are implemented by using a GeoJSON file, in order to have a faster loading time, and this is required because this type of buildings are loaded even when the realistic ones are loaded.

Realistic buildings HVB (presenting photorealistic rooftop details and eventually facades textures) can be loaded as both SceneGraph and 3D tiles. In order to implement the picking functionality we need also to render the extruded buildings underneath. The Extruded type is totally described in a single GeoJSON file, where the following elements are defined for each building: the base polygon, the height, and various other attributes and information. The GeoJSON file is loaded in a layer called GeoJSONLayer, and it is responsible to take all the features in the file and display them on the map, with the base polygon extruded by its height. In the case of Realistic building data type, we use the glTF and GLB formats to describe the scene, and they are loaded by the SceneGraphLayer. This type of integration works well to achieve impressive visualization without impacting too much on the application performances. 3D buildings can also be individually picked on map, in order to see all the building information, besides linking to dedicated BIM representations or other details, if available.

V. Production Process

In this section, we present details of our implemented subprocesses to (A) extract roof patterns, (B) create 3D building with flat roof and photorealistic textures, and (C) integrate HVB and 3D building into a unique 3D representation.

A. Roof pattern extraction

Orthomaps of the city of Florence, kindly provided by the "Sistema Informativo Territoriale ed Ambientale" of Tuscany Region was used to obtain the roof's textures. These RGB photos are tiles with a resolution of 8200x6200 pixels, with partial overlap and rough geo-localization in the EPSG 3003 (Monte Mario / Italy zone 1) coordinate system.

To start with, the aerial images and the 2D GIS building shapes (expressed in the EPSG 4326 coordinate system (Geodetic Parameter Dataset, Originally created by European Petroleum Survey Group)) were converted into a common coordinate reference system. We noticed that by merely translating the orthographic photos from EPSG 3003 to EPSG 4326 was not convenient, as it produced evident alterations in the Ground Sample Distance (GSD, i.e., is the distance, in meters, between pixel centres measured on the ground). To mitigate this effect and better maintain the GSD, we selected a third common coordinate system (EPSG 3857 – WGS84/Pseudo-Mercator) onto which to project both images and shapes.

Multiple orthomap tiles describing the considered area were fused into a single mosaic image using the Geospatial Data Abstraction Library, GDAL (https://gdal.org/). Then, we down sampled the mosaic image by a factor of 1/4. This size reduction was crucial in order to obtain a relevant speed-up in the successive steps, yet without losing accuracy, as the chosen image resolution allows the rooftop detection and alignment deep net (see hereafter) to operate optimally.

To detect the rooftops from the orthomaps and align them with the building shapes, we used the method presented in [23], based on a double U-Net architecture exploiting multi-resolution [25] and multi-task learning [26]. The net takes as input an RGB orthomap and the corresponding cadastral map (represented as a binary image), and outputs a list of multi-polygons aligned with the RGB image. In order to obtain the cadastral map, the 2D shapes of the buildings were converted into a raster binary image. The output multi-polygons, up-scaled to take into account the image down-sampling

previously done, were then exploited to both extract rooftop textures (from the full resolution mosaic) and align them with the 2D building shapes. An affine transformation to warp the mosaic Orthomaps and register it w.r.t. the 2D building shapes was computed. However, using a single transformation for all the multi-polygons would give rise to local inaccuracies. For this reason, we computed a dedicated transformation for each multi-polygon and locally warped the image so as to obtain a better registration. Specifically, given the vertexes of an aligned multi-polygon V_A and the vertexes of the corresponding 2D shape V_S an affine transformation T was estimated such as

$$V_S = TV_A \quad (1)$$

Then, according to the estimated T, the orthomap was warped and the considered rooftop was extracted. After repeating this process for all the multi-polygons, a complete warped orthomap (including only the rooftops) was obtained and exported as JPEG file. Note that, while exporting the texture image, different resolution can be used to obtain smaller weights and faster visualization.

B. *Creation of 3D model with flat roof and photorealistic textures*

3D model construction and texturing were carried out with *Blender*. The building 3D models were obtained by extrusion from the 2D shapes exploiting their height attributes (included in a GeoJSON as above described) with the *BlenderGIS* library. Then a UV-map of the roof areas was created by retrieving the surfaces with normal vectors perpendicular to the main plane, and the warped orthomap was used to texture the polygons described in the UV-map using the *Python Blender API*.

C. *HVB integration*

Using *Blender*, we were also able to include and geo-locating in the map the 3D models of HVBs. For example, as shown in **Figure** 3, an accurate 3D reconstructions of Santa Maria del Fiore Cathedral (Florence Dome) was placed into the 3D representation, thus achieving a nicer final result.

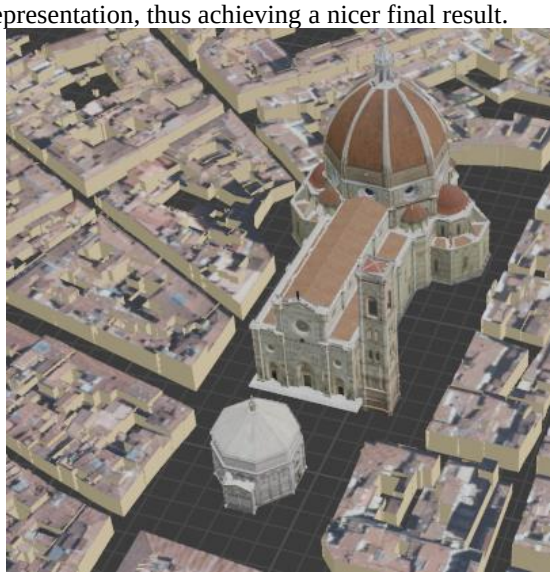

Figure 3: An example of integration of a HVB into the 3D map (in this case the Santa Maria del Fiore Cathedral in Florence).

The obtained 3D textured models of the buildings as well as the HVB models were exported in **glTF** format (including 3D geometries, textures, and coordinates) ready to be deployed in the Snap4City platform using the *SceneGraphLayer* of the *deck.gl* framework (https://deck.gl/).

VI. Access and Distribution in Snap4City

Snap4City is an open-source platform developed at DISIT Lab, University of Florence (https://www.snap4city.org/), [30] [31], [32]. The platform manages heterogeneous data sources, such as: IoT devices (city sensors and actuators, as well as private devices, supporting a large variety of brokers and protocols), open data, external services. For each different kind of data, static attributes (such as geographical information and other metadata) and also real-time data (when available) are collected. Device data are semantically indexed in an RDF Knowledge Base, thus they can be retrieved by dedicated APIs and exploited by Data Analytics processes and IoT applications to perform analyses, simulations, forecasts etc. This allows users to produce new knowledge on data, which can be shown on user interface through Dashboards and a wide range of widgets (showing data both in pull and push modalities). The purpose of integrating the photorealistic 3D city model obtained with the method described in Section IV into the Snap4City platform is to provide a Multi-Data map which can allow the visualization of an interactive 3D environment of the city, with the possibility of inspecting the different kinds of entities and related data, such as: IoT devices, Points of Interests (POI), heatmaps, geometries related to bus routes, cycle paths, traffic flows, etc. In this way, the Snap4City platform allows to exploit a complete open-source framework that can collect, process, and manage all the data needed to obtain a high-fidelity Smart City Digital Twin.

In order to integrate the 3D representations in the Snap4City platform, the deck.gl open-source library has been used, as described in Section V. By exploiting the multi-layer structure of deck.gl, we implemented a distinct layer for every type of data supported by the platform. All layers can be viewed and removed dynamically by user choice. An example of the resulting 3D map is shown in **Figure** 4: the 3D representations can be instantiated by users as a customizable widget in their own dashboards. **Figure** 4 represents the 3D city representation with the addition of textures and 3D model enriched with the textures obtained using the method described in Section V, the model presented in Section IV and the whole architecture of Section III. The tool is freely accessible on web and also includes heatmaps, traffic flow sensors, traffic flow data, animations, PINs for IOT and POI, etc.

Regarding the implementation in deck.gl, first an IconLayer was implemented to represent all the IoT devices managed by the platform. IoT devices are ingested and stored in a semantic Knowledge Base, and they are classified by semantic categories. Therefore, a pool with different icons for each type of device category is used to represent device markers on map. The user can access to all information given by a specific sensor and city element by simply clicking on the device PIN; in this way, a popup is shown presenting static attributes and, when available, real-time and historical data can be selected and viewed on dedicated time-trend and single-content widgets.

Figure 4: 3D Multi Data Map of Snap4City with addition of textures and mesh based 3D building (the Florence dome) [31], [32].
https://www.snap4city.org/dashboardSmartCity/view/index.php?iddasboard=MzI5Mw== accessible to all.

The code of the open source Snap4City Dashboard Builder is available at the following GitHube repository: https://github.com/disit/dashboard-builder.

VII. Rooftop Extraction Validation

To obtain a quantitative validation of the rooftop extraction results on our data, we manually created a set of ground-truth multi-polygon for 200 buildings scattered uniformly on the covered area. Then we evaluated the Intersection over Union (IoU) between the ground-truth and the input (non-aligned) and the output (aligned) multi-polygons.

In **Figure** 5, a bar plot showing the IoU score obtained for each considered building is reported. As can be seen, for almost all the test cases (only in four cases the input multi-polygons have higher IoU), the IoU increases using the output multi-polygons, confirming the effectiveness of the used approach. In average we obtain an IoU score of 0.7100 for the input multi-polygons, and 0.8854 for the output multi-polygons after align them using the deep network, with an increase of almost 17.5%.

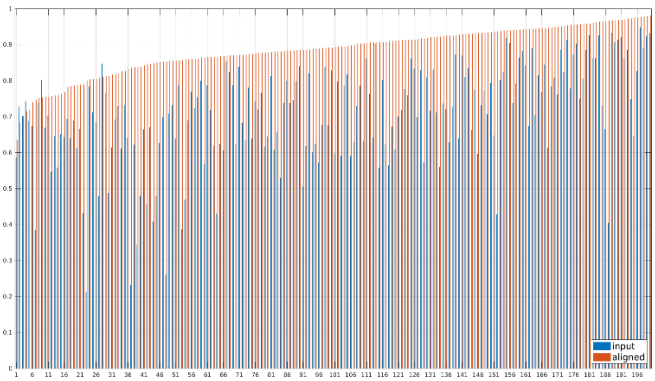

Figure 5: IoU scores for each of the 200 considered buildings. In blue the scores of the input (non-aligned) multi-polygons, in red the results on the output (aligned) multi-polygons. As can be seen, IoU increase for almost all the buildings on the aligned multi-polygons: only in four cases the input multi-polygons obtained better IoU. Note that results are ordered w.r.t. the aligned IoU scores for better readability.

VIII. Conclusions

In this paper, a system for implementing a 3D city model with photorealistic texture integrated into a Smart City framework has been presented. The proposed solution follows a deep learning approach based on U-Net to detect the rooftops from aerial images and align them with the 3D map buildings, which are obtained by extrusion from GeoJSON data. The solution is implemented in the open-source Snap4City platform as a multi-layer 3D map, which can be used by users as a widget on dashboards to visualize a full 3D city environment and a large variety of data, including IoT devices (city sensors and actuators, as well as private devices), POI, heatmaps, geometries and polylines related to cycling paths, bus routes, traffic flow etc. Specifically, users have the possibility to pick on map the single city elements and device markers and inspect their data and attributes. In this way, the proposed solution aims at providing an easy and smart navigation of the global digital twin of the city and the related data. The method employed for rooftop detection and alignment was validated against a set of 200 ground-truth multi-polygons extracted from aerial images of buildings uniformly scattered in the metropolitan area of Florence: after the alignment the, IoU score rises from 0.7370 to 0.8848, confirming the validity of the used approach. As a future work, an automatic procedure is going to be developed, in order

to apply photorealistic texture also to building facades. Many other architecture details have been omitted for the lack of space such as the details regarding the content distribution, the production of facades, the exploitation of Lidar data.

ACKNOWLEDGMENT

Authors would like to thank the HeritData Interreg project. Snap4City (https://www.snap4city.org) is an open technology and research by DISIT Lab, University of Florence, Italy..

REFERENCES

[1] K. Chaturvedi, A. Matheus, S. H. Nguyen and T. H. Kolbe, "Securing Spatial Data Infrastructures for Distributed Smart City applications and services," Future Generation Computing Systems, vol. 101, pp. 723-736, 2019.

[2] N. Lafioune and M. St-Jacque, "Towards the creation of a searchable 3D smart city model," Innovation & Management Review, vol. 17(3), pp. 285-305, 2020.

[3] G. Gröger and L. Plümer, "CityGML Interoperable semantic 3D city models," ISPRS Journal of Photogrammetry and Remote Sensing, pp. 16-21, 2012.

[4] E. Shahat, C. T. Hyun and C. Yeom, "City Digital Twin Potentials: A Review and Research Agenda" MDPI, pp. 3, 2021.

[5] D. Jovanovic, S. Milovanov, I. Ruskovski, M. Govedarica, D. Sladic , A. Radulovic, and V. Pajic, "Building Virtual 3D City Model for Smart Cities Applications: A Case Study on Campus Area of the University of Novi Sad," ISPRS International Journal of Geo-Information, pp. 16-21, 2020.

[6] Helsinki 3D city model. Available online: https://kartta.hel.fi/3d/#/

[7] ETH Zurich VarCity project. Available online: http://www.varcity.ethz.ch/

[8] Bonczak, B.; Kontokosta, C.E. «Large-scale parameterization of 3D building morphology in complex urban landscapes using aerial LiDAR and city administrative data.» Comput. Environ. Urban Syst. pp. 73, pp. 126–142, 2019.

[9] F. Xue, W. Lu, Z. Chen and C. J. Webster, "From LiDAR point cloud towards digital twin city: Clustering city objects based on Gestalt principles," ISPRS J. Photogramm. Remote Sens. pp. 167, pp. 418–431, 2020.

[10] G. Cheng and J. Han, "A survey on object detection in optical remote sensing images," ISPRS Journal of Photogrammetry and Remote Sensing, vol. 117, pp. 11-28, 2016.

[11] J. A. Thompson, J. C. Bell and C. A. Butler, "Digital elevation model resolution: effects on terrain attribute calculation and quantitative soil-landscape modeling," Geoderma, vol. 100, pp. 67-89, 2001.

[12] Y. Ye, J. Shan, L. Bruzzone and L. Shen, "Robust Registration of Multimodal Remote Sensing Images Based on Structural Similarity," IEEE Transactions on Geoscience and Remote Sensing, vol. 55, pp. 2941-2958, 2017.

[13] M. Izadi and P. Saeedi, "Automatic Building Detection in Aerial Images Using a Hierarchical Feature Based Image Segmentation," in 2010 20th International Conference on Pattern Recognition, 2010.

[14] G. Mountrakis, J. Im and C. Ogole, "Support vector machines in remote sensing: A review," ISPRS Journal of Photogrammetry and Remote Sensing, vol. 66, pp. 247-259, 2011.

[15] M. Belgiu and L. Drăguţ, "Random forest in remote sensing: A review of applications and future directions," ISPRS Journal of Photogrammetry and Remote Sensing, vol. 114, pp. 24-31, 2016.

[16] H. Baluyan, B. Joshi, A. Hinai and W. Woon, "Novel Approach for Rooftop Detection Using Support Vector Machine," ISRN Machine Vision, vol. 2013, p. 11, December 2013.

[17] M. Bosch, Z. Kurtz, S. Hagstrom and M. Brown, "A multiple view stereo benchmark for satellite imagery," in 2016 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), 2016.

[18] Y. Zhong, A. Ma, Y. soon Ong, Z. Zhu and L. Zhang, "Computational intelligence in optical remote sensing image processing," Applied Soft Computing, vol. 64, pp. 75-93, 2018.

[19] L. Ma, Y. Liu, X. Zhang, Y. Ye, G. Yin and B. A. Johnson, "Deep learning in remote sensing applications: A meta-analysis and review," ISPRS Journal of Photogrammetry and Remote Sensing, vol. 152, pp. 166-177, 2019.

[20] M. Chen and J. Li, "Deep convolutional neural network application on rooftop detection for aerial image," ArXiv, vol. abs/1910.13509, 2019.

[21] K. He, G. Gkioxari, P. Dollar and R. Girshick, "Mask R-CNN," in Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017.

[22] R. Castello, A. Walch, R. Attias, R. Cadei, S. Jiang and J.-L. Scartezzini, "Quantification of the suitable rooftop area for solar panel installation from overhead imagery using Convolutional Neural Networks," Journal of Physics: Conference Series, vol. 2042, p. 012002, November 2021.

[23] N. Girard, G. Charpiat and Y. Tarabalka, «Aligning and Updating Cadaster Maps with Aerial Images by Multi-task, Multi-resolution Deep Learning,» in ACCV, 2018.

[24] O. Ronneberger, P. Fischer and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Cham, 2015.

[25] A. Zampieri, G. Charpiat and Y. Tarabalka, "Coarse to fine non-rigid registration: a cain of scale-specific neural networks for multimodal image alignment with application to remote sensing," ArXiv, vol. abs/1802.09816, 2018.

[26] S. Ruder, "An Overview of Multi-Task Learning in Deep Neural Networks," ArXiv, vol. abs/1706.05098, 2017.

[27] Rotterdam 3D. Available online: https://www.3drotterdam.nl

[28] Berlin 3D, 3dcitydb. Available online: https://www.3dcitydb.org/3dcitydb-web-map/1.7/3dwebclient/index.html?title=Berlin_Demo&batchSize=1&latitude=52.517479728958044&longitude=13.411141287558161&height=534.3099172951087&heading=345.2992773976952&pitch=-44.26228062802528&roll=359.933888621294&layer_0=url%3Dhttps%253A%252F%252Fwww.3dcitydb.org%252F3dcitydb%252Ffileadmin%252Fmydata%252FBerlin_Demo%252FBerlin_Buildings_rgbTexture_ScaleFactor_0.3%252FBerlin_Buildings_rgbTexture_collada_MasterJSON.json%26name%3DBrlin_Buildings_rgbTexture%26active%3Dtrue%26spreadsheetUrl%3Dhttps%253A%252F%252Fwww.google.com%252Ffusiontables%252FDataSource%253Fdocid%253D19cuclDgIHMqrRQyBwLEztMLeGzP83IBWfEtKQA3B%2526pli%253D1%2523rows%253Aid%253D1%26cityobjectsJsonUrl%3D%26minLodPixels%3D100%26maxLodPixels%3D1.7976931348623157e%252B308%26maxSizeOfCachedTiles%3D200%26maxCountOfVisibleTiles%3D200

[29] Stockholm Opencities Planner. Available online: https://eu.opencitiesplanner.bentley.com/stockholm/stockholmvaxer

[30] Nesi, Paolo, et al. "An integrated smart city platform." Semanitic Keyword-based Search on Structured Data Sources. Springer, Cham, 2017.

[31] P. Bellini, F. Bugli, P. Nesi, G. Pantaleo, M. Paolucci, I. Zaza, "Data Flow Management and Visual Analytic for Big Data Smart City/IOT", 19th IEEE Int. Conf. on Scalable Computing and Communication, IEEE SCALCOM 2019, Leicester, UK https://www.slideshare.net/paolonesi/data-flow-management-and-visual-analytic-for-big-data-smart-cityiot

[32] E. Bellini, P. Bellini, D. Cenni, P. Nesi, G. Pantaleo, I. Paoli, M. Paolucci, "An IoE and Big Multimedia Data approach for Urban Transport System resilience management in Smart City", Sensors, MDPI, 2021, https://www.mdpi.com/1424-8220/21/2/435/pdf

[33] C. Badii, P. Bellini, A. Difino, P. Nesi, "Sii-Mobility: an IOT/IOE architecture to enhance smart city services of mobility and transportation", Sensors, MDPI, 2019. https://doi.org/10.3390/s19010001 https://www.mdpi.com/1424-8220/19/1/1/pdf