IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 2 May 2024, accepted 27 May 2024, date of publication 29 May 2024, date of current version 5 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3406795

RESEARCH ARTICLE

Smart City Digital Twin Framework for Real-Time Multi-Data Integration and Wide Public Distribution

LORENZO ADREANI, PIERFRANCESCO BELLINI, MARCO FANFANI, PAOLO NESI, (Member, IEEE), AND GIANNI PANTALEO

DISIT Laboratory, Department of Information Engineering, University of Florence, 50121 Florence, Italy

Corresponding author: Paolo Nesi (paolo.nesi@unifi.it)

ABSTRACT Digital Twins are becoming fundamental tools to monitor the status of entities, predict their future evolution and simulate alternative scenarios to understand the impact of possible changes for planning and design. More recently, Digital Twin solutions have been applied in the context of Smart Cities. Thanks to the large deployment of sensors, together with the increasing amount of information available for municipalities and governmental organizations, it is possible to build wide virtual reproductions of urban environments including structural data and real-time information that can undoubtfully help decision makers to face future challenges in urban development and improve the citizens' quality of life. In this paper, the Snap4City Smart City Digital Twin framework is presented, which can respond to the requirements identified in recent literature and by international forums. The proposed architecture provides an integrated solution for data gathering, indexing, computing, and information distribution, thus realizing a continuously updated digital twin of the urban environment at global and local scales for monitoring operation and planning. It addresses 3D building models, road networks, Internet of Things entities, points of interest, paths, as well as results from analytical processes for traffic density reconstruction, pollutant dispersion, predictions, and what-if analysis for assessing impact of changes, all integrated into a freely accessible interactive 3D web interface, enabling stakeholder and citizen participation to city decision processes. As case study, the digital twin of the city of Florence (Italy) is presented, including what-if analysis. The solution is released on top of the Snap4City platform as open-source and made available through our GitHub repository (https://github.com/disit) and as Docker compose.

INDEX TERMS 3D city model, 3D web interface, digital twin, Internet of Things/Internet of Everything, ontology, smart city.

I. INTRODUCTION

The concept of digital twin was first introduced in 2002 by Grieves [1], [2] originally named Mirrored Spaces Model [3]. During the years the idea was refined and renamed, till J. Vickers of NASA coined the term Digital Twin [4]. A Digital Twin can be described as digital replica of a physical entity. First applications of the Digital Twin concept can be found in the aircraft and aerospace industry. More formally, as reported in [5], "*A Digital Twin is an integrated multiphysics, multiscale, probabilistic simulation of an as-built vehicle or system that uses the best available physical models, sensor up-dates, fleet history, etc., to mirror the life of its corresponding flying twin*". Later, Digital Twins began to emerge in other fields, such as, for example, in manufacturing [6], [7], or construction industries [8]. The increased adoption of Digital Twin was enabled by technological advances and new paradigm such as Internet-of-Things and Web-of-Things (IoT/WoT) [9], Big Data [10], and Industry 4.0 [11]. Indeed, the information gathered from

The associate editor coordinating the review of this manuscript and approving it for publication was Lukasz Wisniewski.

sensors and devices, and the capability to manage and process huge amount of information [12] are key elements in the development of faithful Digital Twins.

More recently, the concept of Digital Twin has started to be applied in the context of cities [13], in particular for Smart Cities environments and tools [14]. Indeed, according to recent studies on urbanization [15], [16], [17], most of the worldwide population lives in urban areas, while projections indicate that urban population will grow in the near future. It is undoubtful that such increasing urbanization will pose several challenges to decision makers to guarantee satisfactory quality of services for city users. Such challenges span in several fields, from mobility and transport organization [18], [19] up to environmental and energy management [20], [21], as well as to urban planning [22].

In this context, motivations to develop actually Smart City Digital Twins (SCDTs) are a fundamental tool required to both assess the status of urban environment and perform analysis and simulations to guide future tactic and strategic developments and plans. Initially planned to be only as data model without 3D representation, then with the progressive adoption of digital twins by urban planners and designers, they have expressed the need of having 3D representations of the model to assess the impact of changes at level of design in the existing landscape, including simulation for the ground elevation, shading of the buildings, pollutant propagation within the city, exploiting roads and taking into account urban greenery, impact of city decors in walkability, impact of restructuring activities in terms of quality of life, traffic, pollutant, etc. According to these motivations and corresponding requirements, a great part of the research initially has been focused on the construction of 3D City Models [23], three-dimensional representation of the city that can have different Level of Details (LoD) [24]. Clearly, since such 3D models must cover city-wide areas, the production of the 3D models, together with their handling and processing is a challenging task still to be solved [25]. However, a SCDT has to go beyond, not being limited to 3D representations of an urban area at local and global scales, and including real-time data, enabling different kinds of operation such as simulations and predictions for planning, and helping to increment the involvement of citizens though participatory processes [26]. Therefore, SCDT can be used by several user kinds from decision-makers, planners and city officers who can exploit the digital twin as a decision support system in the context of planning short terms changes and mid/long terms plans for city restructuring, making substance cheaper and more realistic simulations of changes. At the same time, they can use the same instrument to dialogue with common non-expert citizens, who can observe the status of their cities and be involved in specific processes, as for example to vote which project they prefer to renovate a public place.

In [13], three main layers composing a SCDT were proposed: a first layer included the heterogeneous data types such as buildings, maps, data from sensors, etc., realizing the so-called City Information Model (CIM). The second layer aimed to cover basic functionalities and communications and it can implement analytics and software. This can be achieved with an adequate IoT/IoE platform able to manage and manipulate huge amount of data. The third layer is charged with the visualization and distribution by means of 3D engines and web applications. In order to guide any SCDT design and development, several requirements were identified in [27]. Those requirements were divided into three main groups: Data, Interactivity, and Interoperability. The first group provided an answer to the question about which data a SCDT must include. Such data encompassed any3D representation of the city such as buildings and other urban entities, together with representations for sensors, services, heatmaps, specific areas and paths. Terrain elevation is considered, too, since it provides information as to building ground positioning, and orographic aspects of the city and its surroundings, thus allowing the effective simulation of the views in the contextual environment. The second group of requirements focused on interactive functionalities that the SCDT interface must provide to become an effective tool for decision makers, being fitted with simulation capabilities. Starting from essential 3D map controls (to allow changing point of view), such functionalities included picking and manipulation activities which are fundamental to provide a valid user experience. Finally, the third group described requirements on interoperability that should be met to guarantee accessible, integrated, replicable, and affordable SCDT solutions, able to ingest data from different sources and with different formats, as well as to provide data analytic services. The above-mentioned requirements are aligned with the main technical challenges, as reported in [26], while they show some limits in supporting the most advanced features needed for urban planning and decision makers: data accuracy and availability are fundamental to represent all the required information in a SCDT, also considering the dynamicity and the amount of data gathered; interoperability aspects are considered at levels of data formats, software, and compatibility of systems; free licenses should be preferred to guarantee the maximum accessibility and avoid technological lock-in on proprietary solutions.

In this paper, the Smart City Digital Twin framework developed on top of the Snap4City platform [28] is described. The motivations behind the design and development of the proposed solution are connected with an effective need by urban designers and decision makers of tools which can actually simulate in depth possible effects of planned changes within the city, while taking into account 3D shapes in the context, impact on traffic, impact of new building in the pollutant, sun shading in the environment, microclimate changes, etc. As stated in [13]: "*Despite increased collaboration between the private and public sector and massive investments in digital twin infrastructures, a 'one-stop-shop city twin' solution ready to be purchased*

from software vendors and implemented on the city scale is yet to be found". For this reason, our aim was to propose an integrated approach capable to respond to as many as possible challenges highlighted in the literature and spanning from data interoperability issues, to appropriate graphical interfaces able to guarantee high level of interactivity and accessibility. Moreover, the proposed solution has been released with open-source licenses. Therefore, it can be adopted by any municipality, thus avoiding technological lock-in on proprietary solutions. To achieve such an ambitious goal, we exploited the Snap4City IoT platform and we integrated into it a novel solution designed and developed for distributing data in a web-based 3D interface. This meant to address several problems related to the huge amount and heterogeneity of data to be displayed and to the specific functionality implemented in order to obtain a solution able to offer better performances with respect to state-of-the-art solutions used to display 3D data on web browsers. As will be better discussed in the following sections, current SCDT solutions have limited capabilities, manage few data kind, do not support interoperability functionalities, and do not implement analytics for simulations and what-if analysis. Differently, our framework solves the above-described limitations exploiting the Snap4City platform and providing an interactive 3D web-based interface to inspect, assess, and study with simulations and predictions the urban environment.

As to this paper, its main contributions are focused to satisfy the needs and requirements identified by decision makers and urban planners, beginning with Snap4City platform. They are:

- a novel high performance digital twin 3D data model distribution and rendering engine which permits the access to complex 3D representations from regular browsers. The new modelling and distribution engine is based on the concept of FusionLayer offering better performance, reducing server requests and lowering client resource usage. It replaced the former architecture used in [27].
- specific functionalities making the 3D digital twin actually functional for the needs of decision makers and urban planners: (i) what-if analysis support; (ii) individual 3D building model selection and substitution; (iii) traffic flow density representation in real time and animations; (iv) heatmap representation and animation in the integrated layer; (v) BIM (Building Information Modelling) management; (vi) road graph visualization as interactive elements.
- an extended set of requirements for smart city digital twins organized in three main groups, namely Field Interoperability, Data and Computing for Representation, and Distribution and Interaction. The proposed requirements are much wider and more precise with respect to the ones proposed in the past in [24] and [27]. They also formalize any actual needs of urban designers and decision makers.
- a wide State of the Art analysis and comparison describing the most relevant SCDT solutions and assessing their level of fulfilment with respect to the identified requirements.

The paper is organized as follows: Section II presents and discuss such defined requirements. Section III illustrates its related works. In Section IV the SCDT general architecture is reported together with descriptions on data modelling, focusing on semantic modelling and Snap4City/Km4City ontology. Section V presents data and processes used to obtain reconstructions, predictions, and simulations and their usage in the SCDT. Sections VI is devoted to illustrating the SCDT distribution through an open interactive web application implementing the 3D rendering engine. In Section VII, the case of study related to the city of Florence in Italy is presented. Section VIII focusses on discussing performance analysis, scalability, and adaptability of the proposed solution. Finally, conclusions are drawn in Section IX. This research has been performed in the context of CN MOST, the National Center on Sustainable Mobility in Italy [29], and for Tourismo Interreg European Commission project on 7 major city/areas in Europe.

II. REQUIREMENT ANALYSIS

To guide both design and development of the proposed SCDT framework, a wide extension and reorganization of requirements presented in [27] is introduced in this section, following indications and needs emerged from recent literature and international forums [13], [26] where several challenges are highlighted without a clear formalization of requirements. Tables 1, 2, and 3 list the defined requirements, providing an ID, a name, and a brief description, they have been extended in the context of wide national meetings of CN MOST, and by the international forum of Tourismo Interreg, which adopted Snap4City as reference platform. The identified requirements have been classified into three groups.

Field Interoperability (see **Table 1**) where requirements are classified as priority in: *Must to have* (M) and *Nice to have* (N) in the last column. These requirements refer to the needs for data ingestion/gathering, data transformation, data storage and modelling, IoT network management and interoperability, exploitation of API, federation of API and platforms, and how to produce data to act on the physical counterpart of the digital twin. Indeed, interoperability is one of the main necessities that an urban digital twin has to support to be able to acquire data from multiple and different sources and keep the digital replica aligned with its physical counterpart [30], [31]. This forces to define appropriate models for network devices and services (**R1**) and entities to be included in the digital twin (**R2**) [32]. Due to data heterogeneity, transformation logics are required (**R3**). To improve interoperability, the adoption of consolidated data models (**R4**), as the FIWARE smart data model [33], and the availability of protocols and APIs to federate different smart

TABLE 1. Requirements for smart city digital twins: field interoperability.

ID	Name	Description	P
R1	Network Modeling and Management	The modeling of data networks including gateways, brokers, devices, services and API, external services as web pages, and protocols.	M
R2	Hierarchical modeling of entities	To model data for terrain, city building and shapes (gardens, roads), services, heatmaps, traffic flows, services, IoT devices, public transportation, etc. To be retrievable with relational, geographical, and temporal queries.	M
R3	Logics for data transformation	To transform data collected, for example from IoT sensors, and other sources and transform them into different data models and formats. For example, collecting data from some web services, GIS, FTP and process them for interoperability and for ingestion.	M
R4	Smart Data Model compatibility	To guarantee interoperable and replicable Smart Cities, interoperability at level of data formats, FIWARE Smart Data Models, etc.	M
R5	Smart City Federation	Model and data federation among platforms at level of protocols and APIs	N
R6	Integration with workflow management systems	To enable ticket/event management. For example, when a fault is detected, it is highlighted in the SCDT and linked to a CMMS (Computerized Maintenance Management System).	N

cities are needed. Finally, to properly handle planned events and emergencies, the integration of workflow management systems is required, for example to plan the activities of maintenance teams (**R6**) [34].

Data and computing for representation (see **Table 2**). This group refers to the needs of modelling and integrating complex data, and its related massive computing for higher level data types that includes reconstruction and predictions for traffic flows, heatmaps, trajectories, 3D representations, alarms, origin destination matrices, etc. The foundation element of a SCDT, such as it should appear to an urban designer and decision maker, should be a 3D city model [35] including terrain information (**R7**), buildings of the city (**R13**), also with BIM representations [36], usually provided in IFC (industry foundation classes) format and conforming to specific standards, such as ISO 19650, and easily generated from any BIM tool used by urban designers. To improve realism and achieve a full CIM [37], ground information (**R8**), e.g., road graphs, specific areas (**R10**), and additional entities (**R15**) must be considered [13]. Then, the CIM has to be augmented with punctual information for services (**R12**), like sensors, POIs (Points of Interest), etc. [38], and heatmaps (**R9**) to observe current and future conditions of the urban environments [39]. Since these data kinds are typically not accessible by third parties, they have to be directly computed on the platform (**R11, R14**) to produce data and insight that are represented in real time in the front-end, by exploiting free license software to avoid technological lock in [40].

Distribution and interaction (see **Table 3**). Former data and computed representation need to be prepared and distributed in a suitable manner to be accessible in the front-end where the SCDT is presented to decision makers and urban planners, and with limited capabilities to a wide public, too. Implemented interactions have to include the capability to change the point of view (**R19**), to select entities to access additional information (**R21.i, R21.ii, R22, R26**) [41], to load different models (**R21.iii**) [42], and to act on lighting (**R20**) [43]. To provide more immediate interpretation of visualized data, entities such as PINs (to indicate POI, devices, dehors, etc., on maps), shapes, paths, etc., should be represented with characteristic visualizations (**R16**) [44]. At the same time, any user should have the possibility to choose which data to display, therefore independent element management is required (**R23**). Since a digital twin must continuously update its data [26], such data have to be automatically rendered in the interface (**R17**) without requiring any reloading (**R18**). To give the user the possibility to request the execution of analytics, business logic call-back has to be supported (**R25**). Finally, digital twin should promote social inclusion and citizen engagement in urban planning [45], [46], therefore freely accessible web-based interfaces are required (**R24**).

III. RELATED WORKS

SCDTs are quite novel, however some solutions have recently begun to appear due to the increasing interest in the topic with public or private commitments. In this Section, a selection of the most representative solutions is reported, only to give a general context on SCDT development endeavors. Then, a deeper analysis has been conducted to assess their compliance with respect to the identified requirements reported in **Tables 1, 2, and 3** and described in the previous section.

Efforts to produce SCDTs have been undertaken by several cities around the world [13]. For example, Helsinki [47], [48], [49], Gothenburg, Paris, Rennes [50], London, New Castle, Rotterdam [51], Berlin [52], Stockholm [44], Zurich [53], Herrenberg [54], Munich in Europe; Toronto [55], Boston [56] in Nord America; Amaravati, Singapore [57], Shanghai [58] in Asia; Wellington [59] in Oceania. Most solutions seem to be work-in-progress or limited case of studies, not providing any open access application to inspect and test their digital twin solutions. Some of them would require the usage of dedicated 3D engines with specific tools installed on high-performance clients, and high-performance hardware on server, making them unsuitable for city user engagement. For example, the SCDTs of Wellington and Shanghai are developed using the Unreal Engine [60]. Such solution requires high-end hardware to render the 3D models on a web browser, thus limiting its distribution.

TABLE 2. Requirements for smart city digital twins: data and computing for representation.

ID	Name	Description	P
R7	Terrain information and elevation	Terrain elevation must be taken into account to properly elevate city buildings and to model city hills and surrounding mountains	M
R8	Ground information	Road shapes and names, names of squares and localities, etc., exploiting orthomaps, with possible real aerial view patterns, and the graph road.	M
R9	Heatmaps	To be superimposed (with variable transparency) on the ground level without overlapping buildings, to represent distribution of temperature, pollutant, noise, humidity, vegetation, etc.	M
R10	Paths and areas	To be used to describe perimeters/shapes of gardens, cycling paths, trajectories, borders of gov areas, elements of origin destination matrices, traffic flows, people flows, trajectories, pipes, sewers, etc.	M
R11	Data analytic	Data analytic processes must be available to let the user develops and/or execute specific data analytics: prediction, traffic flow reconstruction, anomaly detection, pollutant propagation.	M
R12	Single Services	To mark the positions of services, IoT Devices, Point of Interest (POI), Key Performance Indicator (KPI), moving devices as fleets, etc.	M
R13	Buildings of the city	Each single building should be represented. Multiple LoD could be included: (i) simple LoD1 structures, or (ii) higher LoD structures represented as 3D meshes, and (iii) BIMs	M
R14	Automated 3D building construction	(i) 3D buildings must be created automatically, to be able to scale and replicate the SCDT framework; and (ii) the used software must be released with open or free license.	M(i), N(ii)
R15	Additional 3D entities	To augment the realism of 3D representations. For example (i) trees, benches, fountains, semaphores, and any other city furniture, and (ii) water bodies to better represent rivers, lakes, etc.	M(i), N(ii)

TABLE 3. Requirements for smart city digital twins: distribution and interaction.

ID	Name	Description	P
R16	Dynamic 2D/3D structures	Elements such as PINs, shapes, paths, should be represented in 3D dynamically, changing color and shape according to their kind or some real-time value.	M
R17	Dynamic data management	To have elements to be automatically reported in the SCDT as soon as they are included in the platform, event driven rendering of data.	N
R18	No reloading	Changes in the SCDT must be rendered without the needs of a full reload of the map.	M
R19	View Map controls	To change the point of view by zooming, rotating, tilting, and panning the scene.	M
R20	Dynamic sky and lighting	To model and show different sky conditions and to change the light source position, simulating different times of day/night.	M
R21	Building picking/manipulation	To select single building to: (i) show detailed information, or (ii) move into a BIM view of the building, or (iii) to change the building 3D model.	M
R22	Services and element data access	To show data associated with IoT Devices, POI, KPI, shapes, paths, etc., including real time and historical data.	M
R23	Independent element management	To hide, show, replace specific elements (e.g., to disable the building view to see only the city PINs, or to load different heatmaps or paths)	M
R24	Web player	The SCDT must (i) be accessible thought a web browser without additional plugins, and (ii) the player must be released with open or free license.	M(i), N(ii)
R25	Business logic call-back	To provide the possibility of selecting an element (3D, PIN, ground, heatmap) to provoke a call back into a business logic tool for intelligence activities, analytics, etc.	M
R26	Underground and elements inspection	To provide the possibility of selecting and inspecting specific areas and see detailed 3D elements placed underground, such as water pipes, metro lines, etc.	N

For these reasons, hereafter a more detailed analysis focusing only on those SCDTs for which a web interface is freely available was reported. In **Table 4**, a comparison among the inspected solutions is provided considering their compliance with the defined requirements, and colors refer to the previous Tables. The analysis was primarily conducted by investigating and testing the functionality offered through their accessible web interfaces, from which the fulfilment of the requirements on Data and Computing for Representation and Distribution and Interaction can be evaluated. Differently, to assess compliance with the Field Interoperability requirements, available literature was studied. For a better understanding, we have reported in the following text the IDs of the satisfied requirements (from R1 to R26).

The SCDT of the city of Helsinki includes a 2D and 3D interface where is it possible to load LoD3 [24] building models together with terrain elevation data (covering R7 and R13.ii). Each building can be picked (R21.i) to access to additional information, and paths to delimit specific areas can be visualized (R10). Different orthomaps can be loaded (R8) and PINs are displayed mostly to indicate POIs and services (R12 and R22). However, it seems mostly a static representation, since there is no integration with IoT data or other kinds of real-time city related information. We have not found detailed description on interoperability and even if some simulations are described in [48], they did not seem to be integrated in the web interface. The solution proposed by the city of Rotterdam exploits multiple

TABLE 4. Comparison of platforms for SCDT. The (∗) indicates that the functionality is implemented, yet not supported by wide examples. Row colors are related to the colors used in Tables 1, 2, and 3 representing the three requirement groups.

	Helsinki [49]	Rotterdam [51]	Berlin [52]	Stockholm [44]	Zurich [53]	Boston [56]	Snap4City (our)
R1	No	No	No	No	No	No	Yes
R2	Yes (PostGIS)	No	No	No	Yes (Geoportal)	No	Yes (Km4City)
R3	No	No	No	No	No	No	Yes
R4	No	No	No	No	No	No	Yes
R5	No	No	No	No	No	No	Yes
R6	No	No	No	No	No	No	Yes (*)
R7	Yes	No	No	Yes	Yes	Yes	Yes
R8	Yes	Yes	Yes	Yes (fixed)	Yes	Yes (fixed)	Yes
R9	No	No	No	No	No	No	Yes
R10	Yes	Yes	No	Yes	Yes	No	Yes
R11	No	No	No	No	No	No	Yes
R12	Yes	No	No	Yes	No	No	Yes
R13.i	Yes	Yes	No	No	Yes	No	Yes
R13.ii	Yes (LoD3)	Yes (LoD3)	Yes (LoD2)	Yes (LoD3)	Yes (LoD2)	Yes (LoD2)	Yes (LoD3)
R13.iii	No	No	No	No	No	No	Yes
R14.i	Yes	Yes	No	Yes	No	No	Yes
R14.ii	Non free	Non free	n/a	Non free	Non free	n/a	Yes
R15.i	Yes	No	No	Yes	Yes	Yes	Yes
R15.ii	No	Yes	No	No	No	Yes	No
R16	Yes	No	No	Yes	No	No	Yes
R17	No	No	No	No	No	No	Yes
R18	Yes	Yes	Yes	Yes	Yes	Yes	Yes
R19	Yes	Yes	Yes	Yes	Yes	Yes	Yes
R20	Yes	Yes	Yes	No	Yes	Yes	Yes
R21.i	Yes	Yes	Yes	No	Yes	Yes	Yes
R21.ii	No	No	No	No	No	No	Yes
R21.iii	No	No	No	No	No	No	Yes
R22	Yes	Yes	No	Yes	No	No	Yes
R23	Yes	Yes	Yes	No	Yes	No	Yes
R24.i	Yes	Yes	Yes	Yes	Yes	Yes	Yes
R24.ii	Non-free	Free	Free	Non-free	Limited	Non-free	Free
R25	No	No	No	No	No	No	Yes
R26	Yes (*)	Yes (*)	No	No	No	No	No

LoD building models (R13.i and R13.ii) and additional 3D entities (R15) but does not implement terrain elevation. The ground orthomaps cannot be changed, neither heatmaps nor service information are available. Similarly, as to Berlin, the solution includes pickable LoD3 models and the possibility to hide/show single buildings (R13.ii, R21.i). Cast shadows can be visualized and animated according to the time of the day (R20). However, services, paths, heatmaps and real-time data in general are not implemented. The city of Stockholm has implemented many aspects of the Digital Twin concept, such as services (R12 and R22), LoD1/LoD3 buildings (R13.i and R13.ii), others 3D entities (R15) and dynamic cast shadows (R20). However, the solution lacks in the implementation of heatmaps and real-time sensor readings. The SCDT of Zurich has implemented different versions of LoD2 buildings (R13.ii) that can be selected to show specific information in a popup (R21.i). Additionally, buildings can be shown/hidden on user demand, together with building construction plans (highlighted with a different col-our). Additional 3D entities (R15) are included – i.e., trees, with different details that can be selected by the user – as well as terrain elevation (R7). Paths are used to delimit areas and to highlight street elements with pickable functionalities (R10). Cast shadows are not implemented and different illumination conditions can be used to show the model at different times of the day (R20). The solution implements also a 3D measuring tool and a pedestrian-view modality. As to the Helsinki case, no details on interoperability are given, except notes on the usage of Open Government Data. Although in [53] some climate analyses are described, they are not included

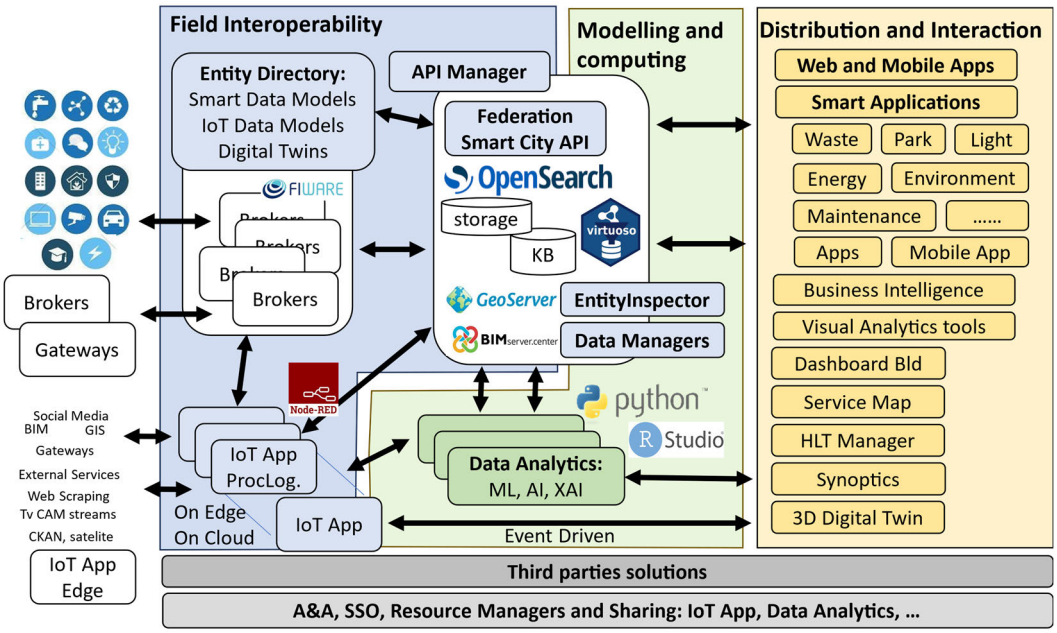


FIGURE 1. Snap4City architecture overview.

in the web interface, neither as heatmaps nor as simulations. As to the city of Boston, LoD2 building models (R13.ii) are visualized and classified into three classes: actual building, building under construction, and approved projects. Trees are represented (R15) and illumination with cast shadows can be changed by the users (R20). Terrain elevation is considered (R7).

To summarize, the compared solutions providing a web interface are closer to simple 3D city representation models rather than to a complete SCDTs. More specifically, some of them do not represent information like heatmaps, services, and paths and no solution seems to be able to aggregate and visualize real-time data, neither to implement analytic processes so as to perform predictions and simulations. Interoperability requirements do not seem to be supported, thus limiting the possibility to deploy such solutions on different scenarios. Differently, the proposed Snap4City solution is a freely available SCDT framework able to model and represent a wide range of data types, show real-time and historic information related to IoT devices and entities, and perform scenario simulations, i.e., What-If analysis, providing the user with a larger set of tools to inspect, assess, change, simulate and study the urban environment. The proposed solution offers a web-based interface to provide access to a wide public to promote and increase citizen participation in the urban development process. At the same time, high level of interactivity is guaranteed. This bridges the gap between state-of-the-art web-based solution offering limited functionalities, and approaches based on 3D gaming engines requiring very high-end hardware on both server and client, thus limiting their distribution. Moreover, it satisfies interoperability requirements, providing efficient solutions for data ingestion and representation, thus facilitating the deploy of the SCDT and guaranteeing its continuous update.

IV. GENERAL ARCHITECTURE AND DATA MODELLING

The presented SCDT framework is built by exploiting the Snap4City platform [61], [62] (www.snap4city.org). Snap4City is an open-source IoT platform coordinated by the DISIT Lab of the University of Florence. Snap4City is an official FIWARE and EOSC platform and it includes a set of Node-RED libraries [63], [64]. It is at present in operational use on several federated installations. The Snap4City platform is able to manage multiple tenants and billions of data with the key focus on interoperability. Snap4City framework is applied in several smart cities and areas in Italy (Firenze, Pisa, Livorno, Prato, Lonato del Garda, Modena, Merano, Cuneo, etc.) and Europe (Antwerp, Santiago De Compostela, Valencia, PontDuGard-Occitanie, Dubrovnik, Mostar, and West Greece, etc.), and their surrounding geographical areas. The largest installation of the platform is a multi-tenant managing advanced smart city IoT/IoE applications with 20 organizations, 40 cities and thousands of operators and developers.

The architecture of Snap4City is reported in **Figure 1**, where the areas mainly addressing the above-mentioned groups of requirements are highlighted. The Field Interoperability and data processing area provide capabilities for collecting data from any sources and exchanging data in push toward any brokers, gateways, and services. This area is interoperable with a large number of protocols and formats (see https://www.snap4city.org/65 for a complete list), compiling with R1, and enabling federation of smart cities (R5) [65].

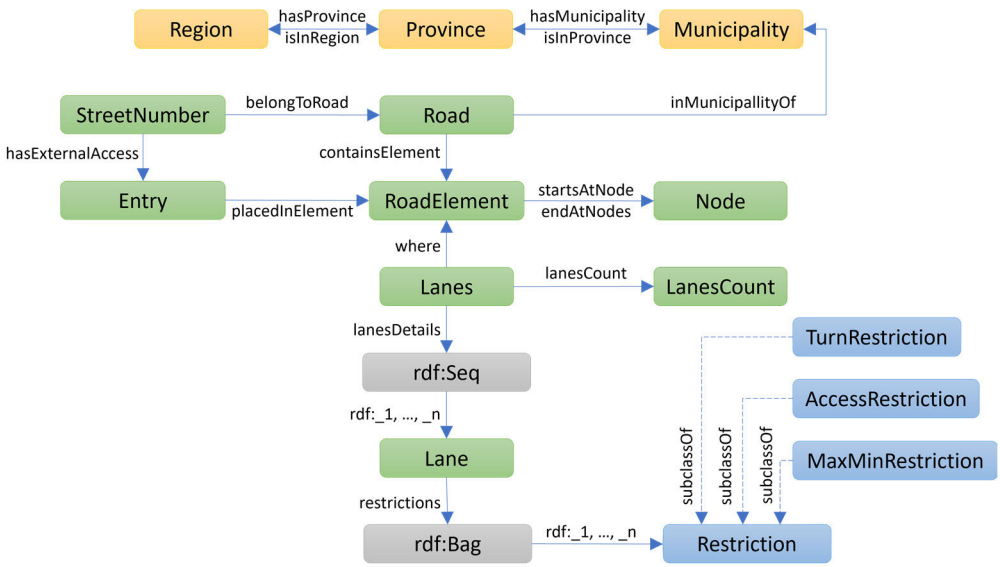


FIGURE 2. Road Graph Data Model (partial) as represented in the KM4City knowledge base.

Real-time data, as well as event driven streams, are ingested using IoT brokers and indexed and shadow stored into an OpenSearch cluster, thus making them accessible to other consumer processes and brokers. Internal brokers are based on Orion Broker NGSI (also compatible with smart data models, R4). Data can be retrieved in push or in an event driven way. Node-RED flows are used to enable data interoperability with third party services such as: GIS (Geographic In-formation Systems), ITS (Intelligence Transport Systems), TV cam services, CKAN open data networks, BIM servers, social media, data gateways, etc. Note that, a specific set of Node-RED libraries and microservices for data transformation (R3) has been developed and made freely available [64]. The collected data are saved into a set of storages (R2): a Virtuoso RDF store handles the Km4City ontology [66] describing semantic relationships among the digital twin entities; an OpenSearch cluster is used to ingest and index real-time and time series data; the Snap4City GeoServer – a GIS server able to provide georeferenced data in tile with user-defined dimensions using the WMS/WFS protocols over HTTPs – to store and provide origin destination matrices, orthomaps, and heatmaps that can describe dispersion of pollutant, temperature, energy, humidity, traffic and people density.

The road graph is represented in the Km4City ontology with entities to describe Roads, RoadElements, Lanes, together with possible Restrictions (see **Figure 2** for a simplified representation) retrieved from OpenStreetMap (OSM). Sensors (IoT as well as Entity devices) are mapped into the Km4City knowledge base using Smart Data Models of FIWARE [67] or other standards, as well as from IoT Device Models of Snap4City. They include traffic, weather, and air quality sensors, as well as other kinds of sensors used to model sharing vehicles, bike racks and parking occupancies, people counting, etc. POIs (Points of Interest) are modelled as Service entities specialized in more than 20 categories (e.g., Accommodation, Cultural Activity, Emergency, Financial, Commercial, Cults), and 500 subcategories.

Building 3D models (see Section IV.B for details on 3D model construction) are stored exploiting a tiled approach following a hierarchical Z/X/Y folder organization used by most of GIS applications like OSM, where Z is the zoom factor (fixed at 18 for our tiles) that describe the tile dimension, while X and Y are the tile coordinates. This solution was adopted to guarantee a fast model loading through the browser interactive web interface (see **Section VI**). The complete city map was divided into non-overlapping tiles and each building was uniquely associated to a single tile considering the centroid of the building shape at the ground. Note that, even if models are grouped in tiles, each single building is represented as a separate entity in order to enable any picking and substitution functionalities (R21).

Finally, the openMAINT Computerized Maintenance Management System, is integrated into Snap4City to handle ticket management for maintenance and operations (R6).

Indeed, the exploitation of the Snap4City IoT platform is a key factor to enable a fast and reliable data acquisition and management, as well as to retrieve any kind of data using dedicated queries and APIs, so to display them on the digital twin interactive web interface, thus responding to the set of requirements on interoperability (R1 – R6).

Data and Computing for Representation and Distribution and Interaction requirements are the most challenging aspects in the development of a SCDT and for this reason the adopted solution is described in depth in the following sections.

V. DATA AND COMPUTING FOR REPRESENTATION

The Data and Computing for Representation group of requirements describes which data need to be addressed in the SCDT and the related computing capabilities to produce them. The computing processes can be classified into two main categories: (i) operative processes used to obtain heatmaps; (ii) generative processes for the construction of the 3D models included in the digital twin. In **Figure 3,** a schematic representation describing the computing processes and their required inputs is reported together with the ingestion processes for IoT sensors, POIs, cycling paths, road graph, and information about public transportations (PA Stops, PA Routes, and PA Schedules).

In the next section, the data represented in the SCDT are introduced, while **Section V-B** is devoted to describing the 3D model production. Details regarding the computation of heatmaps, starting from data stored in the KM4City knowledge base, can be found in [68] and [69].

A. REPRESENTED DATA

At first, a three-dimensional representation of the city is included in order to build an accurate replica of urban structures. This encompasses terrain mesh, building models, and additional 3D entities to augment the representation realism. Terrain elevation is represented by exploiting the RGB encoded Digital Terrain Model (DTM) retrieved from the GeoServer that is used to generate the terrain mesh by means of the Martini tessellation algorithm (R7). Converting the DTM from its original form as matrix of float values to an RGB image is required, since the GeoServer must work with images. In order to obtain the RGB version of the DTM, we have used the following mapping function

$$\begin{cases} R = \left\lfloor \frac{100000+10v}{256^2} \right\rfloor \\ G = \left\lfloor \frac{100000+10v}{256} \right\rfloor - 256R \\ B = \lfloor 100000 + 10v \rfloor - 256^2 R - 256G \end{cases} \tag{1}$$

where $v \in \mathbb{R}$ is the DTM raw float value, while R, G, and B are the image channels. In this way we can obtain an RGB image able to encode elevation differences up to 0.1m. A tiled organization must be exploited to load only visible terrain areas, according to the user's view in the browser. Multiple resolutions are used to model with high precision the urban area, while representing city surroundings using lower resolutions to reduce any possible burden. Each and every following entity included in the proposed SCDT are elevated according to the terrain, so as to avoid sunken or floating elements.

3D building models with different LoD are provided. Simple LoD1 representations (R13.i) and more accurate LoD3 models (R13.ii) are included and can be selected at runtime by the user. BIMs (R13.iii) are accessible from the 3D map using a dedicated interface to perform more in-depth inspections of the building. Dealing with a huge number of buildings, as it is required to represent major cities, is undoubtfully a challenge, in particular when the system must be accessible by any device, without particular hardware requirements. For this reason, specific solutions must be devised. In our case, buildings are organized in tiles at a fixed zoom level and new functions/algorithms have been developed to exploit a tiled representation at multiple zoom levels (see **Section V.C**). Alternatively, multi-zoom grouping could have been exploited; however, such a solution would have required to replicate models in the digital twin storage (to precompute aggregations at different zoom levels), thus ending up in an increase of memory occupancy and a reduction of maintainability, since changes should have been replicated in all the different groups of tiles. Additional 3D entities (R15.i), such as trees or other city furniture may be included in the map using freely available 3D models positioned according to the information provided by municipalities as Open Data. Other entities can be added as well, for example, 3D models of airplanes have been included in the airport area. Appropriate model loading procedures must be used to represent additional entities. Differently from buildings, a limited number of models are required to represent certain kind of entities which are replicated in many positions (e.g., trees belong to 3-4 categories which are replicated hundreds of times with different scale/sizes). Therefore, efficient solutions must be used to retrieve the 3D models from the server once and then replicate them in multiple locations.

The terrain texture is created by merging multiple images. Different orthomaps (R8) can be required by users and loaded at runtime to display specific ground level information (e.g., road map or satellite images). Heatmaps (R9) are superimposed on the orthomap, and can be activated by the user to visualize distributions of temperature, pollutant, noise, humidity, shops, vegetation, etc. Each heatmap is loaded with its corresponding color-map. Heatmaps can be static or animated: static heatmaps are provided as single PNG images, while animated ones are provided in GIF format with multiple images rendered sequentially with a custom delay. Since heatmaps must be superimposed on the ground orthomap, users must have the possibility to set the heatmap opacity. This functionality can be achieved using a multiple image merging process to be able to obtain a single terrain texture that mix orthomaps and heatmaps with different level of opacity (see Section VI-D). Services are loaded from the KM4City knowledge base and represented as PINs to show positions and data of IoT devices, POIs, as for example bus stops (R12), parking, banks, accessible floating bikes, etc. Considering the huge number of services that can be scattered over the city area, efficient functions are required to retrieve and show the associated PINs. Cycling paths and roads, stored in the KM4City knowledge base, are represented as line segments (R10), and are selectable to get more information such as: velocity, category, vehicle kind, status, size, etc.

B. CONSTRUCTING 3D MODELS

As it can be seen from the block-diagram reported in **Figure 3,** the 3D Map construction process starts by

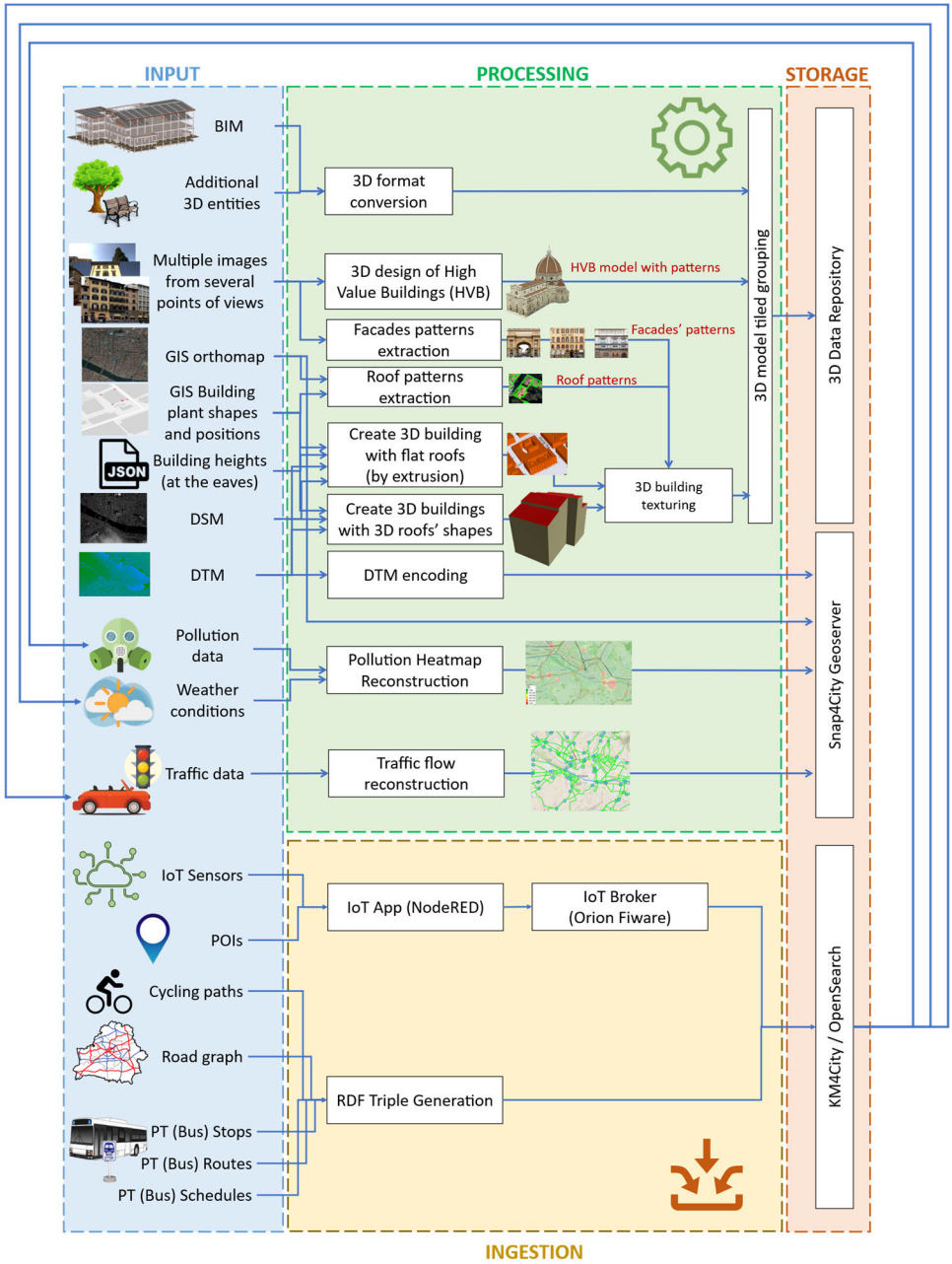


FIGURE 3. Schematic representation of data included in the SCDT. On the left the inputs, on the right the storages. In the middle the generative and operative computing processes used to create the 3D structures and heatmaps which are shown in the upperpart. In the lower part, the ingestion processes used to save data into the KM4City knowledge base and then shadow-copied into an OpenSearch cluster.

gathering different kinds of input data that are elaborated by several processing blocks and finally indexed/archived in specific storages on the Snap4City platform. Inputs include: (i) street-level and aerial (i.e., orthomaps) RGB photos to obtain roof and façade textures and to model possible High Value Buildings (HVB); (ii) building plant shapes from OSM or municipality land registry office (used to geographically localize the buildings); (iii) building height information (in the format of GeoJSON files) and/or Digital Surface Model (DSM) data to properly model the 3D structures; (iv) DTM data to compute a terrain level and to put buildings and other entities at the right elevation and perspective. Moreover, BIMs and additional 3D entities are considered, which may require some format conversion. Hereafter, descriptions of the processing blocks are reported.

1) 3D DESIGN OF HIGH VALUE BUILDINGS

Manual 3D design or automatic computer vision techniques, such as Structure from Motion, can be employed to obtain high quality models for the HVBs. Such models are then put in the right scale, position, and elevation (with respect to the DTM).

2) FAÇADE AND ROOF PATTERN EXTRACTION

Regarding roof and façade patterns, they are respectively extracted from orthomaps and street level images. To obtain an accurate orthomap segmentation, so as to extract the roof texture, a deep net was used [70] to find any similarity transformation required to locally warp the orthomaps and make them accurately fit the building plant shapes [71], [72]. On the other hand, façade patterns are extracted by segmenting the building façade and then rectifying it using planar homographies. Both roof and façade texture are then applied on the 3D building models using Blender plugins developed for these specific tasks.

3) 3D BUILDING CONSTRUCTION

3D structures of ordinary buildings can be obtained with two different approaches. Flat-roof buildings are obtained by extrusion from the building shapes, simply considering the building height. Such height attribute can be obtained from manual measurements of the eave heights, or by evaluating the average height of the DSM samples included into the building plant shape. 3D-roof buildings are obtained by analyzing the DSM and fitting on its samples planar primitives to describe any different roof slopes. Such a process includes spatial clustering [73], [74], multiple linear and planar robust model regression [75]. More details on the 3D-roof building construction are reported in [76]. Both flat-roof and 3D-roof building models can be put at the right terrain elevation exploiting the information encoded in the DTM. The building construction process is carried out automatically (R14.i) and code is freely available (R14.ii) on the DISIT GitHub repository [77].

4) ADDITIONAL 3D ENTITIES

Additional 3D entities (R15.i) such as trees, streetlamps or other minor urban structures/furniture can be obtained from accessible 3D repositories and placed into the map, while exploiting positioning information and size, which can be obtained from Open Data, as well as from municipalities.

VI. DIGITAL TWIN DISTRIBUTION AND INTERACTION

In this section, the complexity of rendering and representing all the SCDT entities (3D models, data, simulations, etc.) is discussed. The first sections are devoted to a general overview, to describe the dynamic 3D representations and the possible interactions, and to present the hierarchical layered structure. Then, in Section VI-C the new Fusion-Layer algorithm, that is able to efficiently manage most of the represented entities, is described. Finally, handling strategies for terrain layer, with orthomaps and heatmaps (Section VI-D), and to support interactivity in What-If analysis (Section VI-E) are presented.

The framework developed for distributing the SCDT as a 3D multi-data map dashboard in the Snap4City platform is able to retrieve and reassemble all the data specified by the requirements from R7 to R15, including: different versions of 3D models of buildings (LoD1, LoD3, BIM) and additional entities, heatmaps (for traffic flow, pollutant dispersion, etc.), services as PINs (IoT, POI, etc.), 3D terrain from DTM, sky pattern, ground information, paths and areas. All the represented entities are localized using the EPGS:4326 (WGS 84) coordinate system. Elements expressed in different coordinate systems can be transformed to the EPGS:4326 during the ingestion and transformation phases. When some analytics require different coordinate systems for their computational domain, transformations are carried out. This approach is not limited to map the ETRS89 (geodetic reference system for Europe), thus allowing the mapping of any kind of data from any countries.

The framework implements a client-side business logic exploiting a series of REST API calls (referred as Smart City API [66]) to load the data independently on user demand (as requested by R23). The 3D building tiled representations are retrieved via HTTPS protocol, as well as data for POIs, IoT devices, paths, etc. obtained with specific geographic queries on the SuperServiceMap of Snap4City, that is GIS interface of the KM4City graph-based RDF knowledge base ontology. Similarly, collected static and real-time data, semantically indexed in the knowledge base, can be retrieved using microservices and/or Smart City APIs to execute spatial, temporal, and relational queries toward the SuperServiceMap. In this way, every time a new element is indexed into the knowledge base it can be automatically displayed on the SCDT interface (R17). Differently, heatmaps (static or animated), orthomaps, and the encoded DTM are retrieved via WMS protocol over HTTPS by querying the GeoServer, limited to the portion of the map visualized by the user.

The SCDT interface has been implemented using WebGL and accessing the client GPU without needing any plugins thanks to the passthrough available in modern web browsers, therefore satisfying R24. Based on the open-source library Deck.gl, the web application has been realized by exploiting a custom implementation and management of the ViewState object, where the geographical information for the map (such as latitude, longitude, zoom, etc.), is defined. Taking into account the viewing position and angle, elements are loaded from the nearest to the farthest with respect to the point of view. This dynamic loading reduces the number of resources needed to display current scene, since elements outside the ViewState are not processed. This is particularly useful when dealing with smart cities covering wide geographical areas. It is worth noticing that all the displayed elements are handled independently and can be shown or hidden at runtime on users' demand (R23) without requiring a full reload of the

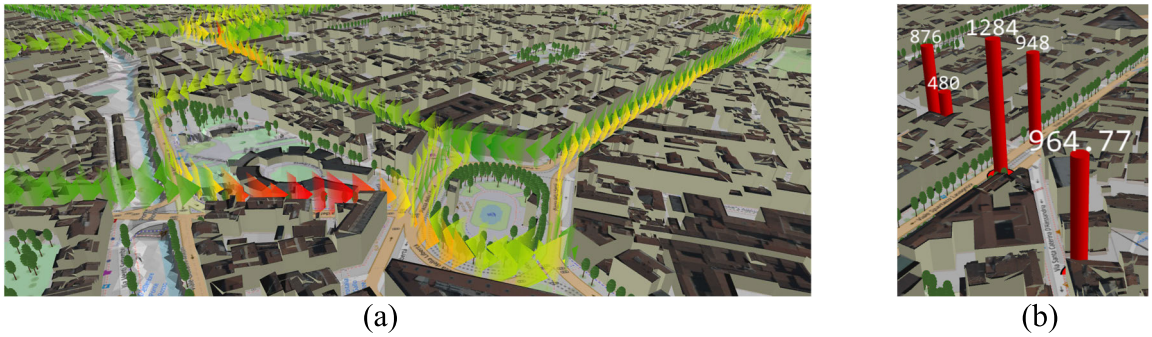


FIGURE 4. Example of dynamic 3D elements: (a) Traffic flow density represented as animated 3D arrows. (b) 3D column representing traffic sensors.

map (R18). To achieve this, the visualization solution was designed to be able to work in a layered way to have different elements represented in specific layers. Finally, resulting 3D representation can be panned, rotated, and zoomed by the user using the mouse or the in-map buttons (R19).

Note that references about accuracies of results concerning analytic processes, i.e., reconstructions, predictions, simulations, etc., usually are not visualized in the SCDT interface, since displaying such information would clog up the interface, reducing its usability. A more appropriate solution would be to have a separate dashboard or specialized widgets to show such data. However, for relevant cases, specific KPIs can be set up with such information and displayed in the SCDT map as interactive PINs.

A. DYNAMIC 3D REPRESENTATIONS AND INTERACTIONS

To better show particular information on a 3D map, which is one of the fundamental characteristics of a digital twin, new kinds of representations have been designed and developed. For instance, traffic flows, typically represented by colored lines over the ground, can be better represented using static or dynamic 3D elements (R16). In our SCDT, traffic flow densities are shown as raised crests (with amplitude proportional to the computed traffic density values) and colored using a color map that can be defined by the user. Alternatively, traffic can be displayed as animated arrows (see **Figure 4a**) moving with a speed related to the real-time traffic density (different animation times are used to represent different traffic conditions, faster for free roads, slower for congested segments). Both crests and arrows follow terrain elevation, and by mouse hovering over the elements, popups are displayed reporting traffic information. Due to the nature of the problem, traffic flow polylines can be fragmented. Then, for each adjacent segment points are passed to the GPU and interpolated to render the crest profile exploiting the fragment shader to obtain a considerable speed-up.

3D columns (see **Figure 4b**) are used to represent real-time data values registered from IoT sensors: the value of any sensors can be shown with a 3D cylinder (or other solid) shape, positioned where the sensor is, and whose height is proportional to the considered metric, while its actual value is reported in textual form on top of the cylinder. The color of the column represents the sensor category. All these elements can be customized by the user.

Additional interactivity is provided for buildings, additional 3D elements, PINs, and paths. Each building model can be selected (R21.i) to access additional information that are reported on a panel in the map. To achieve such functionality, an appropriate in-tile indexing is provided to single out a 3D building from any tile. From the building panel, the user can select different 3D representations of the same structure (R21.iii) to change at runtime the model used in the map and get access to a BIM representation (R21.ii) that is visualized and that can be inspected in a dedicated interface. In **Figure 5**, an example of building substitution is reported. Such a functionality is useful to modify any buildings included into the map, for example to appreciate different possible city construction plans to be shown to a wide public, thus improving citizen engagement towards urban landscape evolution. If the aim is to represent buildings as independent 3D entities, so as to offer picking and substitution functionalities, this poses additional challenges w.r.t. typically implemented solutions based on 3D tiles [78] where different 3D entities are fused in a single mesh. Transmission and rendering of a huge number of single 3D buildings requires long times and reduces interface usability due to lags and general sluggishness. Usually, such problem is solved by strongly limiting the number of visualized elements (for example [49] and [52] display only few buildings at a time). In a quite different way, the proposed FusionLayer (as described in detail in Section VI-C) solves these problems and can represent thousands of buildings.

Information regarding additional 3D entities (e.g., trees), road elements, and cycling paths can be accessed by the user by hovering the mouse on a specific element, so as to bring up popups with additional details (like the street name, the tree species, etc.). Services are represented as 3D PINs with a different icon and color according to their semantic category. After selecting them, a popup window is shown reporting real-time or historic data (R22) – see **Figure 6**. Moreover, by clicking on a sensor measurement type, an event can be

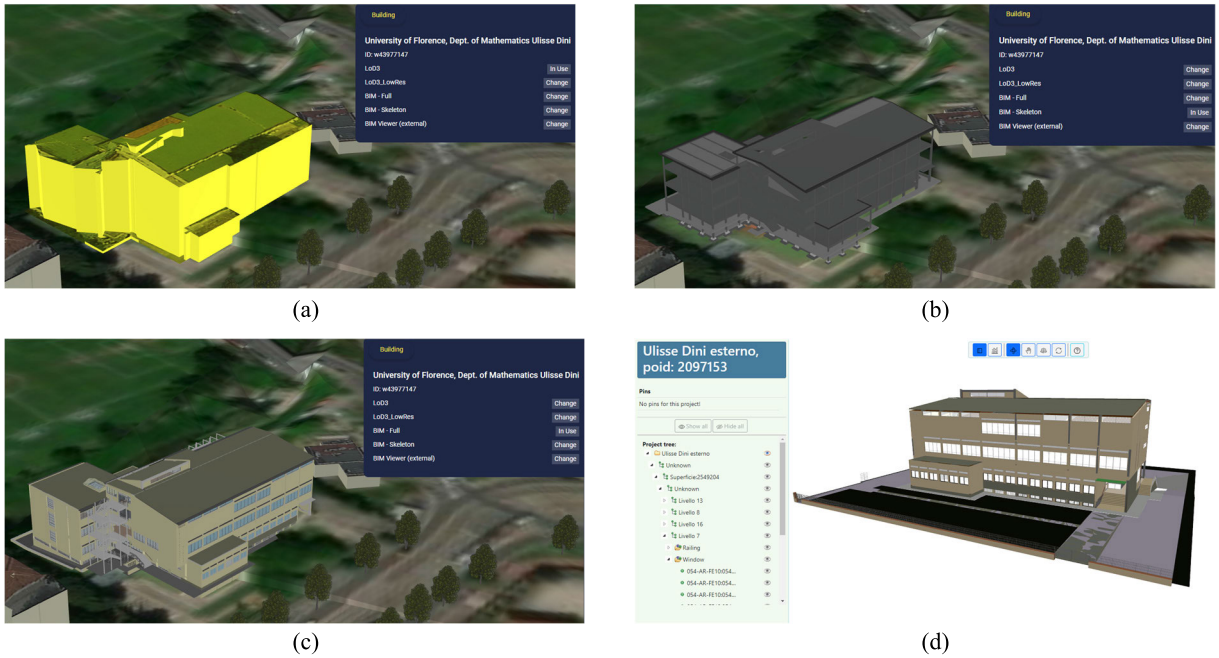


FIGURE 5. Example of 3D building substitution and BIM integration. In (a) the default LoD3 building model is selected (highlighted in yellow) and a panel in the top left corner of the interface shows building related information and other available 3D models. By clicking on the buttons, different models are loaded in the SCDT web interface without requiring any refresh. For instance, in (b) a skeleton structure of the building is shown, while in (c) a GLB model obtained by converting the IFC BIM representation is displayed. Finally, in (d) the full BIM of the building is presented: such a representation is accessible from one of the panel buttons and opened in a dedicated dashboard with additional functionalities for BIM inspection.

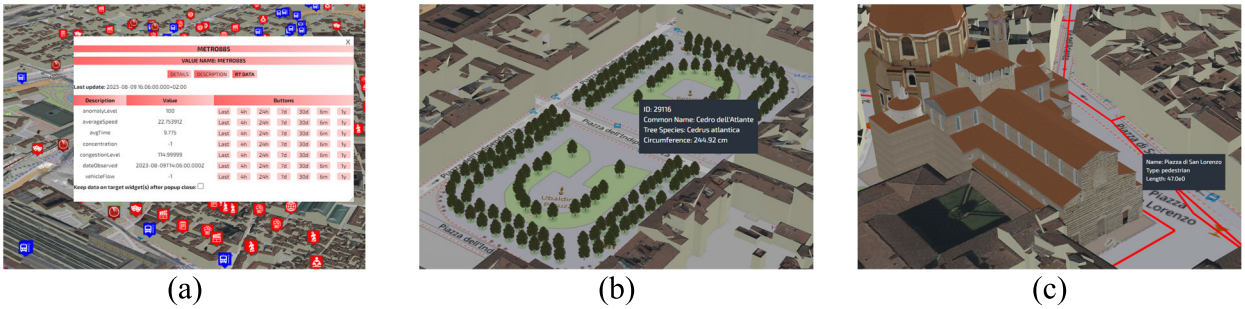


FIGURE 6. Example of interactive pop-up with specific element information for services as PINs (a), additional 3D elements, as trees, (b), and road elements (c).

raised to provoke a call-back to business logic (R25) and to show additional data in separated widgets included in the same dashboard (e.g., time trends) exploiting client-side business-logic functionalities [79]. The possibility to show entities according to their semantic categories and to display real-time and historic data was possible thanks to the adoption of the Snap4City platform to manage data ingestion, semantic indexing, and retrieval. Digital twin solutions which do not exploit similar platforms, as the ones discussed in Section III, are therefore limited, and cannot offer a comparable access to information as the proposed framework.

Scene illumination (R20) has been modelled with two types of lights: an ambient light to affect the entire scene and a directional light to model the position of the sun (calculated according to [80]). This process creates lights and shadows of the scene, and it is useful to simulate when a particular area is either well illuminated or not.

B. HIERARCHICAL LAYERED STRUCTURE

In the previous version of Snap4City SCDT [27] tile loading was based on a hierarchical layered structure mainly exploiting the TileLayer of Deck.gl with some modifications to handle specific data needs. For example, to display buildings at different zoom levels, we have modified the SceneGraphLayer, nested in the TileLayer, in order to handle multiple different 3D models and reduce web traffic and server calls, changing the default Deck.gl behavior. However, such a solution gives rise to problems when dealing with a huge number of tiles. Since some elements to be displayed are organized at a fixed zoom level, when showing wide areas,

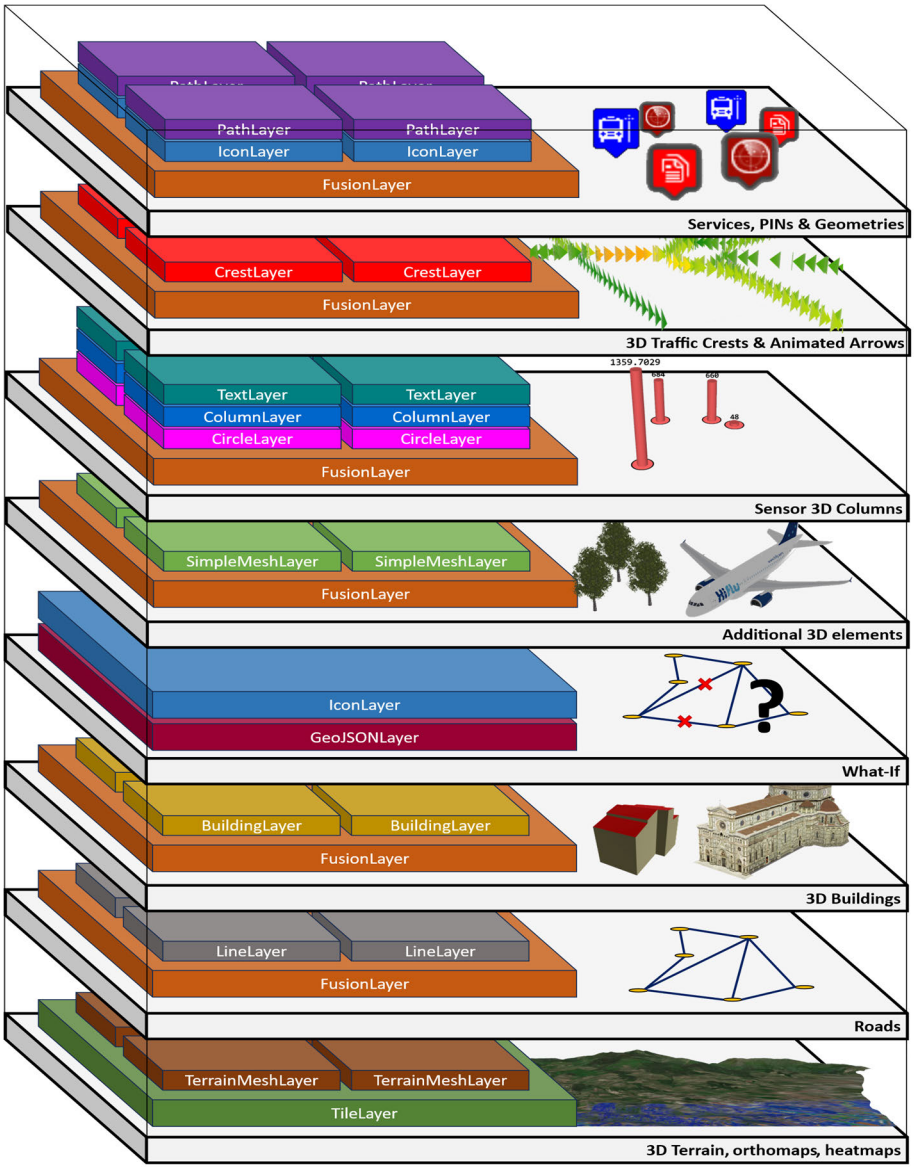


FIGURE 7. Representation of the layered architecture used to develop the interactive web-based interface to represent all the data required in the SCDT exploiting the novel FusionLayer. Further details on the Deck.gl architecture extended with the proposed FusionLayer can be found in Section VI-C and Figure 8.

a large number of tiles must be loaded in the client GPU and CPU, thus augmenting the use of resources and making the user's experience worse, due to lags and general slowness. Additionally, moving to a different area would have required to make new server requests and retrieve anew some entities already previously downloaded.

Moreover, in the previous version, 3D crests and columns, PINs, and paths, were loaded without using a tiled approach to obtain all the entities in a single request, regardless of the zoom factor. Indeed, such elements are always the same at any zoom level (which is something different from heatmaps or orthomaps that could be provided with finer or rougher details according to the zoom) and this solution was adopted to avoid additional server calls. However, such a solution has brought forth a significant slowdown in data rendering, since only after completing the download globally, entities were drawn.

In order to solve these issues, i.e., to improve any tile support and avoid new server requests related to already downloaded elements, the novel FusionLayer was developed and is hereafter described. A graphical representation of the novel layer structure is shown in **Figure 7**. As can be seen, each layer is used to represent a different data kind. This layered architecture has been designed to display in a single interactive interface the multiple data kinds that a digital twin may include. Therefore, the 3D Terrain, orthomaps, heatmaps layer is devoted to representing the 3D meshed terrain textured with orthomaps and heatmaps; the Roads

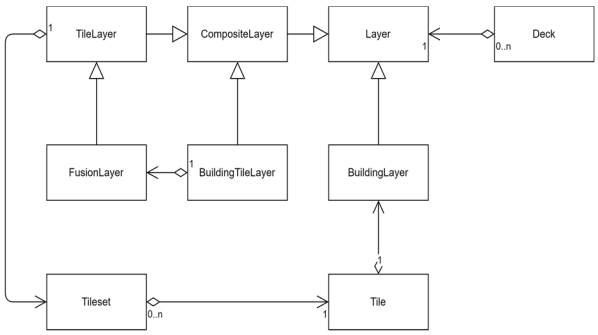


FIGURE 8. UML class diagram describing the updated Deck.gl architecture including the proposed FusionLayer. Being derived from the TileLayer, the FusionLayer can include a Tileset to control multiple Tiles. In this example, each Tile includes the BuildingLayer, that is devoted to representing multiple 3D buildings in GLB format. Further details can be found in Section VI-C.

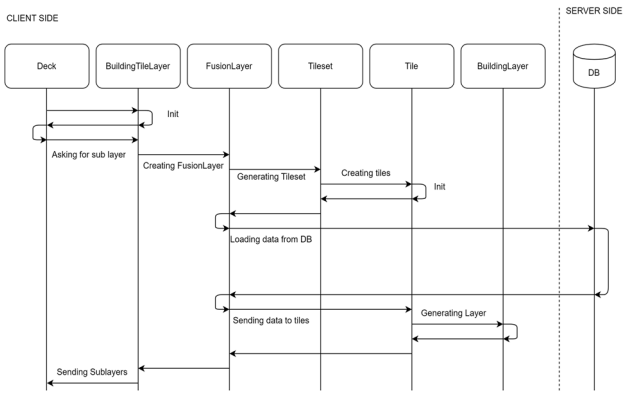


FIGURE 9. Sequence diagram describing the interactions between the novel FusionLayer and other classes of the Deck.gl architecture.

layer is used to show the road graph retrieved from the knowledge base. The 3D Buildings layer is specialized in handling a huge number of 3D models representing the building of the city, while the Additional 3D elements layer is in charge of rendering a few 3D models (e.g., planes, trees) in multiple positions. The What-If layer is used to show the changes in the road graph performed to describe the scenario context of the what-if analysis. The Sensor 3D Columns and the 3D Traffic Crest & Animated Arrows layers have been developed to render the newly devised representations for IoT sensors and animated traffic flow densities, respectively. Finally, the Service, PINs & Geometry is devoted to show PINs for IoT devices and services, and particular paths, e.g., the public transport routes, cycling path, trajectories, etc.

C. FUSIONLAYER

To keep retro-compatibility and exploit some native Deck.gl functions, the FusionLayer was derived from the TileLayer. In **Figure 8**, the class diagram of a simplified representation of the Deck.gl architecture, modified with FusionLayer, is presented. As it is evident, Deck.gl can render multiple kinds of Layer classes, that are abstract classes. Derived from the Layer class, the CompositeLayer (another abstract class) is designed to manage other sub-layers. In the presented example, we have limited the description to the BuildingTileLayer (derived from the CompositeLayer and used to manage the 3D buildings) that only includes the FusionLayer. Similar relations hold for other kinds of layers specialized to handle different entities, i.e., PIN/POIs, geometries, IoT, roads, etc. The FusionLayer, being derived from the TileLayer, is also a CompositeLayer and therefore able to control a variable number of sub-layers. The number of sub-layers to handle is specified in the Tileset class, that is devoted to creating multiple Tiles – that are geo-referenced objects and can contain multiple layers. In this case, only the BuildingLayer is used, a class specialized to handle multiple GLB files representing 3D buildings and implementing all the functionalities described in previous sections.

To further explain both architecture and functionalities of the proposed FusionLayer, in **Figure 9** a sequence diagram is reported. After performing a request to display the 3D buildings over the map, the Deck main object (instantiated when loading the map) creates and initializes the BuildingTileLayer object, then it asks for the required sublayers to be rendered in the view. Once the request is received, the BuildingTileLayer creates the FusionLayer that becomes responsible for the creation of single tiles. At first, the FusionLayer generates the Tileset that is used to build single tiles. After that, the FusionLayer performs a request to the server data repository, it retrieves any required data and sends data to specific tiles. Each tile then creates a BuildingLayer and send results back to the FusionLayer, that in turn forwards them to the BuildingTileLayer and finally to the main Deck object to be displayed to the user.

In practice, we introduced the FusionLayer in order to have a class specialized in handling data in any tiles to be displayed for the different layers included in our multi-layered architecture. Such a solution helps us to provide a more efficient and smarter data management, by reducing data exchanges between the server and the client (in terms of number of requests made and downloaded data) and by facilitating tile control on specific data types, therefore improving the overall system performance. More specifically, the FusionLayer introduces the following additional functionalities: (i) reduces the number of server calls by introducing an element cache management on the client exploited by two main functions bottomUpFusion and topDownFusion; (ii) reduces the usage of client resources by implementing the deepLoadFunction to retrieve elements available only at a specific zoom and then group (or split) results at run-time in tiles with correct size, according to the visualized zoom level, thus avoiding too many tiles to be rendered.

The bottomUpFusion and topDownFusion are two mirrorlike functions devised to reuse already retrieved data and avoid new server calls, see **Figure 10**. More in details, let's suppose to start with a tiled visualization at zoom level z_i and then move to a zoom level z_j, with $j > i$, i.e., the user is

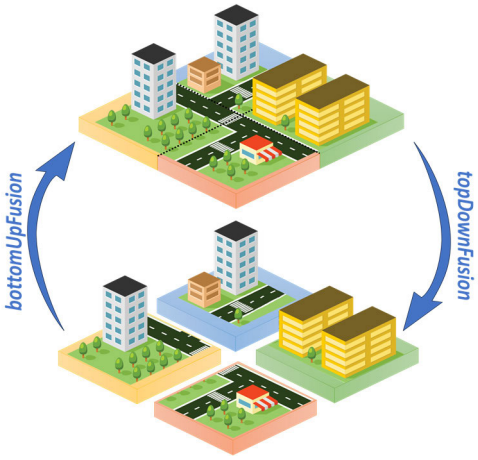


FIGURE 10. Schematic example of the bottomUpFusion and topDownFusion. Using the bottomUpFusion, data from the four child tiles are passed into the parent tile. Conversely, using the topDownFusion, data in the parent tile are transferred to the child tiles.

zooming into the map. The topDownFusion is used. Based on exploiting the hierarchical tile organization, the system knows which tiles at zoom z_j correspond to a tile at zoom z_i. For each child tile t_c at z_j, the system searches for a parent tile t_p at z_i and retrieves the cached elements. Then each element position is checked and if it is assessed that it belongs to the tile t_c, the element is added to the feature array of tile t_c, avoiding any retrieving it again from the server. Similarly, when the zoom level changes from z_j to z_i, i.e., the user is zooming out, the bottomUpFusion is invoked. In this case, all the elements in the chid tiles t_c at zoom z_j are added to the feature array of the parent tile t_p at z_i. For both fusions, if for a tile no data is found in the cache, i.e., the child/parent tile was not previously loaded, then new server calls are executed to retrieve data. This happens when the user moves to different map areas originally not in the bounding box of the current view. To explain the logic behind the use of the FusionLayer, in **Figure 11** a flow chart of BottomUp/TopDown logic is presented. As it can be seen, as a first step the system checks if the requested tile is already loaded, otherwise it checks if there is a parent tile, then if there are any child tiles. If neither parent nor child tiles are available, the system loads the entire tile form the server. Otherwise, data in the parent/child tiles are recovered and loaded in the new tile, thus avoiding unnecessary server calls.

It is noteworthy that both topDownFusion and bottomUp-Fusion can work even if the considered zoom levels are not contiguous (i.e., $j - i > 1$). In such a case, the jumpZoom functionality is exploited to put in relation child and parent tiles of not contiguous zoom levels.

Some elements to be displayed in the web interface are available at a specific zoom level. For example, building models are grouped in tiles at fixed zoom level 18. Using a naïve solution, if the scene in view covers a wide area, a large number of tiles would be loaded in the GPU, thus slowing down the system performance. To solve this problem, the deepLoadFunction has been developed. Let's suppose to have elements available only at zoom level z_j and being in a ViewState condition with zoom level z_i, with $i < j$. The deepLoadFunction initially splits tiles at zoom z_i in n virtual sub-tile at zoom z_j. For each of these sub-tiles any required elements are retrieved with n independent asynchronous requests. Finally, when all the responses are collected, the system aggregates all the data and displays them in a unique layer at zoom z_i, using the correct number of tiles to be displayed and avoiding an excessive number of rendering operations. Data retrieval can exploit also the cached element using the bottomUpFusion or topDownFusion.

The FusionLayer can be specialized to work with different kinds of geo-localized data. For instance, it is possible to define a JSONFusionLayer to work with plain JSON file, and a GeoJSONFusionLayer to deal with GeoJSON data. Nevertheless, the FusionLayer can be easily extended to work with additional data formats, such as SVG, CSV etc.

In the actual implementation, the FusionLayer is employed to load buildings, additional 3D elements (e.g., trees), paths and shapes, PINs for IoT Sensors, POIs, services, and dynamic 3D elements such as traffic crests or animated arrows and 3D columns (as shown in **Figure 7**). For each data kind, a new instance of the FusionLayer is instantiated to load elements independently and with different safe contexts. In particular, using the FusionLayer, buildings are always retrieved at tiles at zoom 18 and displayed at different zoom levels without excessive GPU loads thanks to the exploitation of the deepLoadFunction, and this similarly occurs with 3D crests and arrows. PINs, paths, 3D crest, arrows and columns, are now loaded in a tiled and dynamic modality, enhancing the interface responsiveness, while still limiting the server requests, since already downloaded data could be reused thanks to the fusion mechanism. Elements falling out of the viewport are deleted from any client memory after a given delay to reduce the computational burden.

Moreover, this tiled approach allows the system to perform a more accurate data retrieval. Indeed, most GIS servers can only provide data included in a squared bounding box, that is adequate when working on 2D maps. Conversely, a perspective view of a 3D map typically draws a trapezoid on the main plane. Therefore, to query all the elements in the viewport a bounding box wider of the viewed area is required. Thanks to our new tiled approach, the system can query the server using finer bounding boxes, thus limiting the retrieval of elements outside the viewport, and being able to query bounding boxes with different sizes (i.e., different zoom level), smaller as to tiles being closer to the viewpoint, wider as to those being more distant.

D. TERRAIN, ORTHOMAPS, AND HEATMAPS LAYERS

Not all the data to be visualized in the interactive web interface of the SCDT need to exploit the novel FusionLayer. Terrain data, orthomaps, and heatmaps are provided by the GeoServer in tile of user-specified zoom level and at different resolutions (higher resolutions for higher zooms covering smaller areas and vice versa). In this case the

topDownFusion and the bottomUpFusion cannot be used, since for different levels of zoom new data must be retrieved to obtain the correct resolution. Similarly, there is no need to use the deepLoadFunction, since data can be requested at different zoom levels. This implemented solution can handle a three-dimensional terrain using the default TileLayer to display in tiles the RGB encoded DTM retrieved from the GeoServer. The obtained DTM is used to generate the terrain mesh by means of the Martini tessellation algorithm. It is possible to mix multiple DTM files, for example with different resolutions: in such a case one of the DTMs has higher priority over the others.

As anticipated in Section V, the terrain texture is created by merging multiple images: the base image is the orthomap of the terrain, over which different heatmaps can be shown. Different orthomaps and heatmaps can be loaded on user demand. This data integration forms a layer called TerrainMeshLayer. The texture merging process is carried out directly inside the GPU to have maximum performance. To handle the selected opacity level, the following equations have been used to merge different textures inside the fragment shader:

$$\begin{cases} \text{mix}_{\alpha} = 1 - (1-\alpha_2) * (1-\alpha_1) \\ \text{mix}_{R} = \left(\dfrac{R_2 * \alpha_2}{\text{mix}_{\alpha}}\right) + \left(\dfrac{R_1 * \alpha_1 * (1-\alpha_2)}{mix_{\alpha}}\right) \\ \text{mix}_{G} = \left(\dfrac{G_2 * \alpha_2}{\text{mix}_{\alpha}}\right) + \left(\dfrac{G_1 * \alpha_1 * (1-\alpha_2)}{mix_{\alpha}}\right) \\ \text{mix}_{B} = \left(\dfrac{B_2 * \alpha_2}{\text{mix}_{\alpha}}\right) + \left(\dfrac{B_1 * \alpha_1 * (1-\alpha_2)}{\text{mix}_{\alpha}}\right) \end{cases} \tag{2}$$

where α_i is the alpha channel of the background image ($i = 1$), and the additional image ($i = 2$) to be merged, while R_i, C_i, and B_i are the RGB channels. When merging three or more images, the process is performed progressively for each pair, cumulating the next one on the first pair merged.

A custom rendering system was implemented in order to add the *SkyBox* feature based on a direct access to the WebGL context, required to include a sky representation into the 3D map.

E. MANAGING WHAT-IF LAYER

A key functionality for urban planning and management that can be offered by a SCDT is the possibility to perform simulation and observe the effects produced by a change in the contextual environment. For example, to observe how vehicle routing can change due to a scenario where an area is blocked to traffic. Such a functionality is called What-If analysis [81]. This can be applied to different kinds of problems to understand the impact of different scenarios on traffic flow, routing, pollutant diffusions, people flow, etc. To achieve this, analytic processes (R11) are exploited through business-logic call-backs (R25) to send user inputs and obtain results to be visualized at run-time, without requiring any explicit change on the road graph or map reloading. Both What-If analysis results and selected scenario, represented as elevated lines to be better visualized in a 3D map, are shown to the user through the WhatIfLayer, developed from scratch. The WhatIfLayer is a composite layer formed by GeoJSONLayers and IconLayers that are rendered dynamically as the user interacts with the map. An example of routing What-If analysis implemented in our SCDT is presented in Section VII and in **Figure 12**.

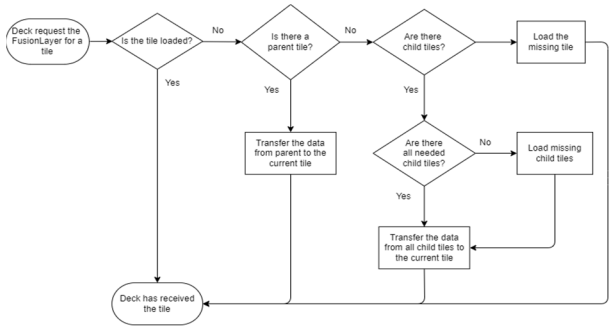


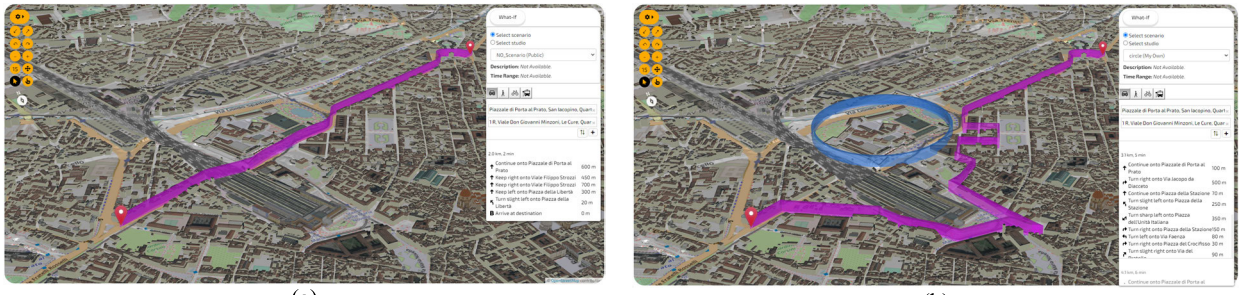
FIGURE 11. Flow diagram describing the algorithm exploiting bottomUpFusion and the topDownFusion functionalities of the FusionLayer.

VII. CASE OF STUDY: THE SCDT OF FLORENCE

In order to validate the proposed Snap4City SCDT framework, we have selected as case of study the city of Florence in Italy. The Florence SCDT encompasses the full municipality (as shown in **Figure 13**) covering an approximate extension of 151 km2 (wider than the area of the Florence municipality of 102 km2). The SCDT of Florence is freely accessible at https://digitaltwin.snap4city.org.

In **Figure 14,** a screenshot of the interactive web interface of the SCDT, implemented as a 3D multi-data map dashboard in the Snap4City platform, is reported. On the left side, a selector menu is displayed to show/hide information coming from analytic service or real-time data. The main portion of the web interface is devoted to representing a 3D scene. In the top-left corner in-map controls are included; the pointing-hand icons can be clicked to activate the picking functionality on buildings as shown in the map to get detailed information, to change the displayed 3D model, and to access to a BIM representation when available. Moreover, the gear icon can be used to get access to an in-map menu where the user has the possibility to select different orthomaps, buildings, and to activate animations, decorations, and road information from a dedicated setting panel. In the top-right corner a panel can be shown to provide the user with additional information and controls. Additional widgets to show time series, histograms, etc., can be included in the same dashboard and updated in real-time using event-driven callbacks to a client-side business logic [82].

Florence SCDT includes LoD1 and LoD3 3D building models and terrain elevation. The DSM and DTM data, used to model respectively buildings and terrain, have been kindly provided by the "*Sistema Informativo Territoriale ed Ambientale*" of Tuscany Region. They have been obtained from a LiDAR survey and are composed by several tiles



(a) (b)

FIGURE 12. Example of What-If analysis on vehicle routing. In (a) the initial routing, shown as a purple elevated line. In (b) a What-If scenario is enabled: a blue shape highlights the blocked area, and the updated routing is shown to the user.

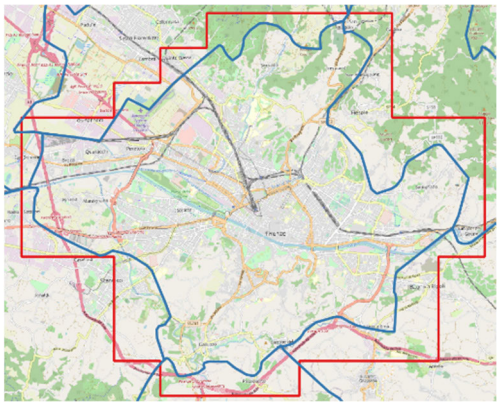


FIGURE 13. Extension of the modelled Florence area. In blue the administrative border of Florence municipality; in red the area covered by our Digital Twin. This map is shown in the EPGS:3003 coordinate system.

covering the city of Florence, with a resolution of 1 square meter. 3D building models have been enhanced with roof textures obtained from orthomaps of the city of Florence. The RGB photos are tiles with a resolution of 8200 × 6200 pixels, with partial overlap and rough geo-localization in the EPSG 3003 (Monte Mario / Italy zone 1) coordinate system. The SCDT includes services represented as PINs indicating positions of POIs, IoT sensors, bus stops, parking, etc. Thanks to the semantic indexing of data offered by the Snap4City knowledge base, different PINs can be represented with specific icons according to their semantic category [83], [84]. Information associated with a specific service can be accessed by simply clicking on the respective PIN: a popup is shown to the user presenting static attributes and, when available, real-time, and historical data. 3D column representations are also available to display city traffic sensors. Moreover, heatmaps, with an opacity level that can be set by the user, are loaded, and superimposed on the ground orthomap at runtime to show real-time distribution of pollutant, temperature, humidity, etc. Traffic densities are shown as animated 3D crests or arrows to be better visualized in the 3D map. The road graph is queried from the knowledge base and shown over the ground terrain to access street information by hovering over the displayed line segments.

To show the capability of our SCDT system, which allows the user to carry out simulation and analysis on routing, controls to run What-If analysis were implemented into the web interface. By using a separate interface – by visiting https://www.snap4city.org/dashboardSmartCity/view/index.php?iddasboard=MjE5MA== and activating the Scenario selector to define a new scenario for What-If analysis – the user can select specific points or areas to simulate a traffic restriction. Then the scenario can be loaded into the SCDT interface (activating the What-If selector), and the routing algorithm produces trajectories between any start and end position considering the defined restriction. An example of What-If on routing is presented in **Figure 12**: as it can be seen, an area was selected to ban traffic from some enclosed streets and the updated routing is shown to the user.

To summarize, the SCDT of Florence includes 3D structures of the urban environment together with static and real-time data describing the status of the city. Simulations are possible by exchanging buildings and performing What-If analysis. Thanks to the developed 3D web engine, the SCDT can be distributed though web browsers, requiring neither additional plugins, nor high-end hardware, thus being accessible from a wide audience and enabling possible coworking in virtual living lab and helping to include also common citizens in the urban evolution process.

VIII. PERFORMANCE, SCALABILITY, AND ADAPTABILITY

A. PERFORMANCE ANALYSIS

The performance analysis has been evaluated by carrying out a number of stress tests to assess the impact of the novel FusionLayer of the proposed SCDT framework. We have compared the proposed solution with respect to the TileLayer, used in our previous version, and which is the state-of-the-art solution proposed by DeckGL to handle tiled data. Tests have been executed on a PC equipped with an Intel Core i9-12900K@3.20 GHz, with 32GB of RAM, and an NVIDIA RTX A4500 GPU, using Google Chrome browser and its built-in network inspector.

We have performed the tests on the (i) Services, PINs, and Geometry, (ii) Roads, (iii) 3D Columns, and (iv) 3D Traffic Crests layers to observe the downloaded data (in MBytes),

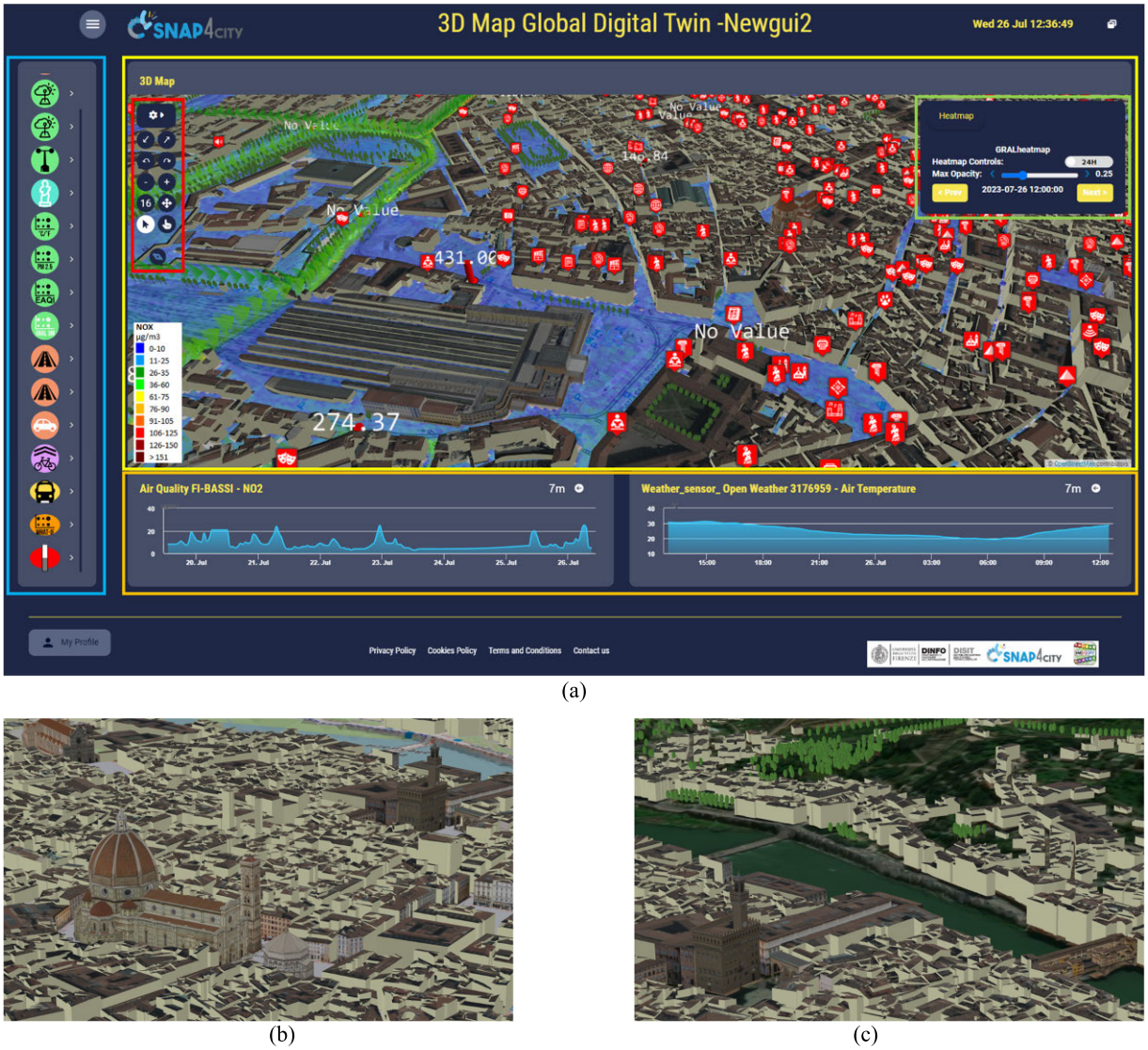


FIGURE 14. Snap4City dashboard showing the Smart City Digital Twin of Florence. In (a) the dashboard is presented with LoD3 and HVB models shown together with PINs, 3D animated arrows for traffic, heatmaps, 3D Cylinders, trees. The blue rectangle highlights the selected menu. In yellow the main 3D multi-data map, with the in-map menu (red rectangle) and the interactive panel (green rectangle). Under the map, in the orange rectangle, additional widget can be visualized. In (b) a close-up view of Florence city center. In (c) another close-up shows entities correctly elevated, according to the 3D terrain (textured with satellite orthomaps). To try our SCDT of Florence the reader is invited to visit the following link https://digitaltwin.snap4city.org.

and the number of requests performed to the server. We did not include in the tests any Terrain or What-If layers, since they do not use the FusionLayer. Moreover, we excluded the 3D Buildings and the Additional 3D Elements layers, since the former was strongly changed with respect to its previous implementation and a fair comparison would not be possible – mostly due to the introduction of the possibility to change any 3D building representation in real-time directly from the web interface – while the latter, mainly used to represent trees, requires to download a single JSON file and a single 3D model, thus not changing at different zoom levels. In **Tables 5 and 6** the results for the transmitted MB and the number of requests are reported respectively.

Tests were carried out with the following protocol regardless of each kind of layer: the tester enters in the SCDT interface; the system loads in the default view state; the tester opens the Chrome network inspector, it clears the possible outputs, and sets a filter to consider only the calls related to the layer under inspection; the tester enables the considered layer and measures the MB and the number of requests done (these values represent the initial state, equal for both the FusionLayer and the TyleLayer, and are reported in the first columns of **Tables 5 and 6**). Then, the tester changes the zoom level to measure the benefit of the bottomUpFusion (zoom-out) and the topDownFusion (zoom-in) algorithms and registers the updated measurements from the network

TABLE 5. Performance analysis for downloaded data (MB) without and with the fusionlayer for services, pins and geometry, roads, 3d columns, and 3d traffic crests. data are in mbytes.

Layers	Initialization	BottomUp		TopDown	
		Without FusionLayer	With FusionLayer	Without FusionLayer	With FusionLayer
Services, PINs, and Geometry	0.442	2.000	0.866	0.135	0.000
Roads	11.400	78.000	50.300	4.000	0.000
3D Columns	0.073	0.328	0.172	0.041	0.000
3D Traffic Crests	3.400	18.500	18.500	0.000	0.000

inspector. When zooming-out, the system passes from zoom level 16 to 14, while as to zoom-in it shifts from level 16 to level 18. It is noteworthy that measurements of MB and the number of requests is deterministically related to the amount of data encompassed into the visible area in the map: MBs depend on the number of entities that must be fetched from the server, while the number of requests depends on the number of visualized tiles. Therefore, such results, even if obtained on a specific area, can be generalized for any position: the more data are included, the greater and more relevant are the benefits of using the FusionLayer. Moreover, since the number of considered tiles are equal for all the tests when using the same method,(i.e., with or without FusionLayer), we could measure, as expected, an equal number of requests for each and every test case (Services, PINs, and Geometry, Roads, and 3D Columns), the only exception being the 3D Traffic layer and this has to be better discussed in the following.

As it can be seen in **Table 6**, the use of the FusionLayer has reduced the number of downloaded data in all the cases, thanks to its capability to avoid fetching anew already retrieved data. As to Services, PINs, and Geometry the volume of downloaded data has been reduced from 2 MB to 0.866 MB in the bottomUpFusion, and from 0.135 MB to 0 MB for the topDownFusion. Similarly, as to Roads, retrieved data pass from 78 MB to 50.3 MB for the bottomUpFusion, and from 4 MB to 0 MB in the topDownFusion. Finally, as to 3D Columns representing values of sensors, downloaded data have been reduced from 0.328 MB to 0.172 MB using the bottomUpFusion, and from 0.041 MB to 0 MB using the topDownFusion. Clearly, the FusionLayer has strongly reduced the downloaded data during zooming, in particular for the zoom-in operation invoking the topDownFusion, since no additional data are retrieved from the server.

In **Table 6**, the number of requests made for the above-described test cases is reported. The topDownFusion provides evident benefits in reducing server requests in all the cases (except the 3D Traffic) passing from 49 to 0 requests. Indeed, during zoom-in operation, since data at zoom level z have been already fetched from the server, when passing to zoom level z' > z, using the FusionLayer no additional requests are made.

Differently, using the TileLayer, when the zoom-level changes and new tiles are formed, new requests are made regardless of whether the data were already fetched or not. For the bottomUpFusion (i.e., zoom-out) few additional requests are performed: 42 without the FusionLayer, 45 with the FusionLayer. This effect is due to the different tile dimensions at different zoom levels: for example, when moving from a zoom-level z to a level z-1, tiles become four times bigger, i.e., four tiles at level z fall in a single tile at level z-1. When using the bottomUpFusion, if concerning only some tiles at level z-1, data were already retrieved when passing to tiles at level z, the system queries the server for such missing data and uses the smaller tiles of level z to fill the new tile at level z-1, thus slightly augmenting the number of requests. However, the actual volume of downloaded data is still lower as shown in the previous experiment.

A final comment must be stressed to the performance of the 3D Traffic layer. As it can be seen from **Tables 5 and 6**, there are no differences whether using or not the FusionLayer. This is due to traffic data being retrieved at a fixed zoom level in order to obtain a sufficient detailed representation, thus requiring the same number of calls and MBs exchanged for both approaches when changing any zoom level (zero request when zooming-in, and 80 requests and 18.5 MB of transmitted data when zooming-out). Nevertheless, using the FusionLayer offers some benefits: while the TileLayer works with tiles of fixed dimension at any visualized zoom, the FusionLayer dynamically changes any tile dimension according to the zoom level and this leads to a computational advantage in the rendering phase. Indeed, each tile is rendered independently, one-by-one and using different tile dimensions at different zoom levels reduces the number of rendering operations to be performed. For example, when zooming-out from level 16 to level 15, tiles become four times bigger and concurrently the same area is covered by four times less tiles, which implies a reduction of the rendering computational burden.

TABLE 6. Performance analysis for number of request calls without and with the fusionlayer for services, pins and geometry, roads, 3d columns, and 3d traffic crests. Data are the number of requests.

Layers	Initialization	BottomUp		TopDown	
		Without FusionLayer	With FusionLayer	Without FusionLayer	With FusionLayer
Services, PINs, and Geometry	25	42	45	49	0
Roads	25	42	45	49	0
3D Columns	25	42	45	49	0
3D Traffic Crests	47	80	80	0	0

TABLE 7. Performance analysis for rendering times without and with the fusionlayer for services, pins and geometry, roads, 3d columns, and 3d traffic crests. Numbers indicate average values with respective standard deviations in brackets. values are expressed in seconds.

Layers	Initialization	BottomUp		TopDown	
		Without FusionLayer	With FusionLayer	Without FusionLayer	With FusionLayer
Services, PINs, and Geometry	1.61 (0.46)	3.12 (0.34)	3.54 (0.95)	0.69 (0.11)	0.45 (0.04)
Roads	10.40 (0.62)	18.50 (1.52)	19.18 (1.91)	0.68 (0.11)	0.66 (0.11)
3D Columns	2.91 (0.88)	9.21 (1.29)	13.20 (9.47)	0.58 (0.12)	0.45 (0.08)
3D Traffic Crests	5.31 (0.73)	34.93 (9.83)	26.40 (5.81)	0.73 (0.15)	0.52 (0.09)

To complete the performance analysis, in **Table 7** the rendering times required to retrieve and draw the entities in the web interface are reported. As to previous tests, the same testing procedure was adopted, using this time the Chrome performance inspector to assess rendering times. In this case, differently from the MB and the number of requests, measurements are not deterministic, and times can vary due to network lags or CPU/GPU loads. Moreover, since manual interactions are required to perform the tests, the tester speed can have some impact on the obtained times. Therefore, we repeated the measurements ten times to obtain average and standard deviation values. As it can be seen from **Table 7**, times are comparable for both solutions, either exploiting or not the FusionLayer. This suggests that the FusionLayer and its data management among tiles at different zoom levels do not significantly impact on rendering times, still granting advantages on data transmissions. The 3D Traffic Crest layer is the layer showing the greatest time differences both for the bottomUp and the topDown, thus confirming the observation discussed in the previous paragraph, namely that the FusionLayer can offer better performance by reducing the number of tiles to be rendered.

B. SCALABILITY AND ADAPTABILITY

The proposed SCDT framework can be easily deployed in other scenarios representing different smart cities, even of bigger size with respect to the Florence use case. On one hand, the Snap4City platform is designed to scale on huge amount of data that can be ingested and handled efficiently thanks to its data modelling and storage capabilities (as described in Section IV). On the other hand, the interactive web interface was designed to be able to represent all the required data, regardless of any smart city dimension, and to be accessed by several users at the same time. This was achieved by exploiting the tiled representation and limiting the visualization of elements to the actual area in view, with a client-side business logic to carry out elaboration directly on the client, thus reducing the impact of multiple user connections on server side. Clearly, to observe the whole city landscape a client with an adequate hardware is required, mainly due to the processing and rendering operations used to display the 3D elements. For this reason, we have granted to the user the possibility to set some parameters to tune the SCDT according to his available hardware: by accepting a simpler visualization with less decorations and graphic details, even a low-end client can still play and use a SCDT built with our framework.

The Snap4City platform, including the SCDT interactive interface, is released as open source. The source code can be obtained from our GitHub repository, as well from a containerized version based on Docker Compose [85]. Therefore, any municipality has the possibility to deploy

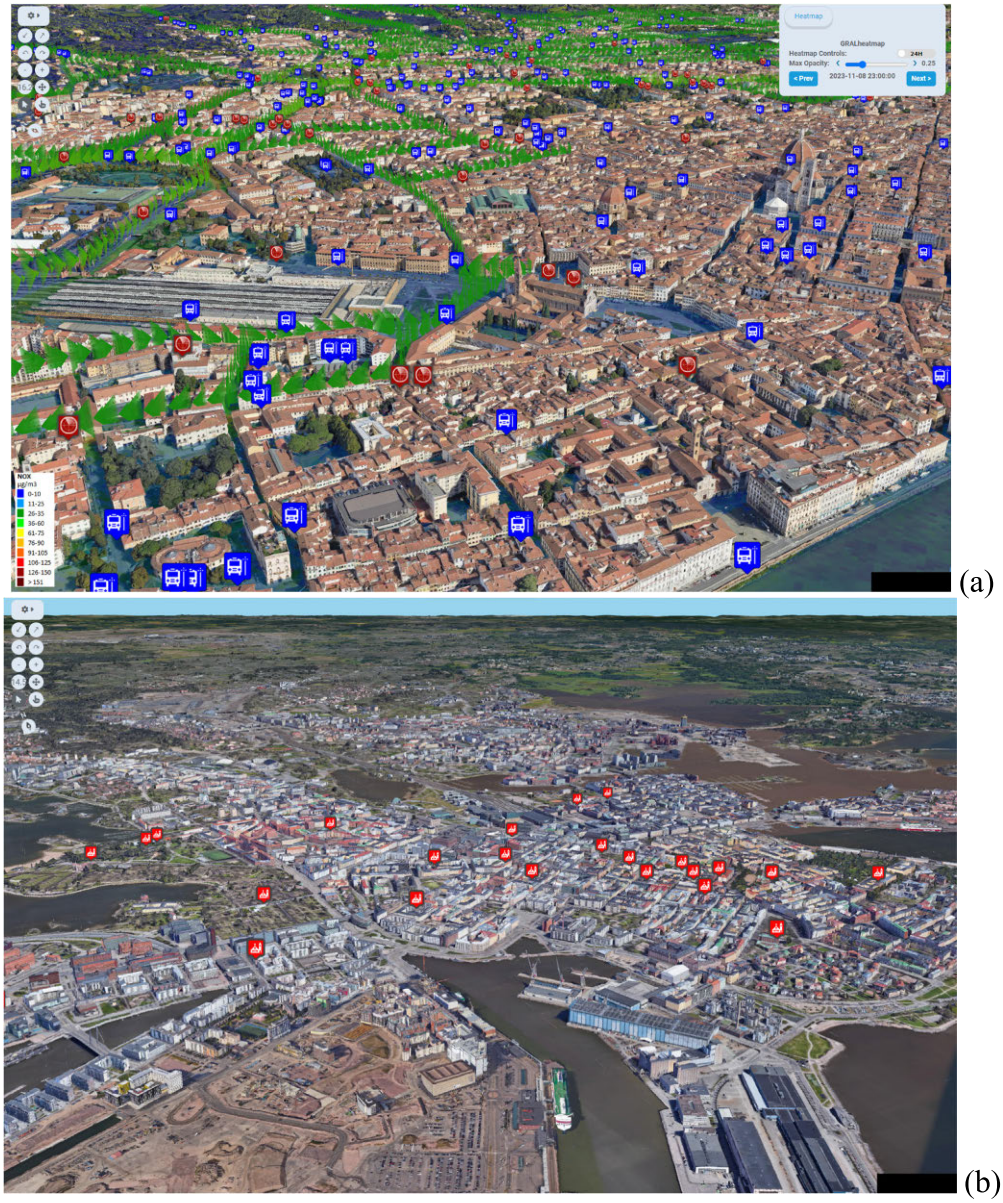


FIGURE 15. Example of integration of Google 3D tiles in Snap4City SCDT. In (a) Florence, Italy. In (b) Helsinki, Finland. Clearly visualized data depend on local availability.

its own Snap4City platform with its related digital twin. According to data availability, IoT models and entities can be created, and real-time data can be ingested through an IoT broker exploiting Node-RED flows. Moreover, analytic services can be used to compute more complex information such as heatmaps or traffic flows. The road graph, including streets, and pedestrian and cycling paths can be obtained from free services, as for example OpenStreetMap, that nowadays covers most cities worldwide.

One of the most complex data to acquire are the buildings' 3D models, since very few municipalities have an available 3D land registry to be exploited. The availability of DSM/DTM data obtained by aerial surveys are more common and when such information is available, the construction process described in Section V-B can be exploited (code is freely available): a full city 3D reconstruction encompassing more than 20,000 buildings (approximately the size of Florence city center) can be obtained in about 2 days of computation. However, since each building model construction is an independent process, parallel execution on multiple cores or even on different workstations/servers could be exploited to significantly reduce the total time, enabling cities with a high number of buildings to be represented in reasonable time. Other solutions, based for example on Structure from Motion approaches, can be considered, and an in-depth discussion on such topic is out of the scope of this paper. If no building related data are available, other services can be exploited. For example, OSM Buildings [86] offer LoD1 models freely for non-commercial solutions, while, more recently, Google released

FIGURE 16. Example of building picking/selection functionality on Google 3D tiles using invisible extruded buildings shapes.

photorealistic 3D tiles [87] that can be exploited to represent city buildings. In this case, the Google 3D tiles are loaded using a Tile3DLayer replacing the Terrain, the 3D Buildings, and the Additional 3D elements layers of our architecture (see again **Figure 7**). All the other entities included into our SCDT can still be represented over the Google 3D tiles effortlessly, thanks both to our layered architecture design which is able to handle any different kinds of data to be displayed, and to the underlying Snap4City Knowledge Base and services that can be queried to retrieve road descriptions, position of POIs, heatmaps, and to perform What-If analysis. As an example, in **Figure 15** we have reported the usage of the Google 3D tiles into our Snap4City SCDT interface for the cities of Florence and Helsinki. In the first case – **Figure 15(a)** – all the elements described in the previous sections (i.e., PINs, 3D Traffic Crests, heatmaps, paths, etc.) can still be represented when using Google tiles, thus demonstrating how the proposed digital twin architecture does not depend on the used 3D building representation. Conversely, in **Figure 15(b)** the city of Helsinki is shown in the proposed SCDT framework, together with some PINs indicating the position of POIs, demonstrating how our solution can be effectively deployed in different municipalities. It should be remarked however that the adoption of Google 3D tiles imposes some restrictions, both on usage licenses (that at the time of our writing are not yet well defined by Google) and on the interactivity, since in 3D tiles different buildings are fused in a single mesh. Indeed, in this case, building substitution is no longer possible without elaborating Google tiles to single out individual buildings (something that licenses do not permit). However, we managed to provide a building picking functionality thanks to a smart solution exploiting invisible extruded buildings used to delimit volumes of single structures (see **Figure 16**).

IX. CONCLUSION

In this paper, the Snap4City Smart City Digital Twin framework has been presented. Indeed, the design and development of a framework able to implement a SCDT that can be deployed by any city with limited efforts is challenging task: specific requirements must be defined, adequate solution for handling different kind of data must put in place, and open, accessible, and interactive 3D interfaces must be developed to let any user access the SCDT information and experiment through simulations and predictions.

According to the needs expressed by decision makers and urban planners we identified a large set of requirements and classify them in: Field Interoperability, Data and Computing for Representations, and Distribution and Interaction. After an in-depth comparative analysis among several smart city digital twin solutions with respect to the defined requirements, our SCDT framework has been described in detail to show its compliance with the defined requirements.

The requirements identified and satisfied by the platform enable stakeholders and decision makers to use the SCDT from web pages for planning and simulation. For those experts the possibility of making interactive changes and see effects in real time is fundamental to understand both context and effects of possible decisions. The usage of simple projects in 2D is not sufficient to represent effective results. For example, pollutant flows, the shadow of buildings in squares, views from the road. The proposed framework was designed and developed to aggregate different kinds of data and seamlessly represent them on a freely accessible interactive web interface, thus realizing an accessible, integrated, replicable, and affordable SCDT solution.

Differently from other state-of-the-art SCDTs, the proposed framework exploits capabilities of the Snap4City IoT platform to ingest, manage, index, and provide through APIs data describing the urban environment that can be continuously updated with real-time information. Our solution exploits interoperable 3D building models and additional 3D entities, together with an accurate terrain model textured with user selectable orthomaps and heatmaps, and information on specific paths and areas (e.g., road graph, cycling paths, animated heatmaps, etc.) to represent the urban infrastructures. Different entities like IoT devices, POIs, and services are represented as dynamic PINs that can be clicked to access additional static, historic and real-time data. Specifically devised 3D representations (i.e., 3D crests, arrows, and columns) are used to show real-time traffic density reconstruction or specific sensors. The proposed system supports What-If analysis tools to modify the actual context producing possible scenarios and simulating the impact of the introduced changes (in terms of traffic flow, routing, and pollutant). The paper has introduced a novel high performance 3D data distribution and rendering engine for digital twins which permits to access an interactive usage/exploitation of the complex 3D SCDT representations from regular browsers. The new rendering engine is based on a layered structure to load all the elements independently, with their own safe context, thus avoiding reciprocal interferences and exploiting a tiled subdivision of the scene. On such grounds, the newly introduced concept of FusionLayer supporting smart caching to deal with huge volumes of data offers better performance reducing server requests and lowering client resource usage, as demonstrated

with experimental tests, thus improving any final user experience. The developed interface can integrate several data of different kinds, using novel client-side business logic. The developed framework was used to implement the SCDT of Florence, Italy, to demonstrate all the devised functionalities. Moreover, we have shown that the proposed solution can be easily deployed on different cities worldwide, thus realizing the only freely available SCTD framework able to represent a wide range of data, perform simulations, and provide the user with a large set of tools to inspect, assess, and study the urban environment. This research has been performed in the context of CN MOST, the National Center on Sustainable Mobility in Italy, and for Tourismo Interreg European Commission project on 7 major city/areas in Europe.

Future works will address the introduction of additional elements and functionalities to further improve both realism and interactivity exploiting the proposed modular and layered architecture of our SCDT framework. For example, streetlamps can be added as additional 3D elements to show the city lighting by nights, underground structures can be introduced, and novel analytics integrated, such as solar panel potential, flooding, or 5G coverage, also in the context of What-If analysis.

APPENDIX
ACCESS TO THE SOLUTION AND ITS SHORT USER MANUAL

The proposed Smart City Digital Twin framework, implemented in the use case described in Section VII, is freely accessible at https://digitaltwin.snap4city.org. However, this public version does not include the Google 3D tiles due to license limitations. You can ask the authors to have a full access exploiting the Google 3D tiles.

In order to experiment the full potential of the SCDT, we suggest using the Google Chrome web browser on a workstation equipped with an Intel Core i9-12900K@3.20 GHz, with 32GB of RAM, and an NVIDIA RTX A4500 GPU (the same configuration used to carry out the performance tests) or a superior configuration. The solution also works quite well with lighter solutions. In the event a user did not have access to this configuration, reduced settings can be used and yet be able to test our SCDT without experimenting sluggishness (see what follows for a description on how to set the preferred parameters).

In Section VII and Figure 13 a brief description of the main components and panels available in the dashboard has been provided: on the left side, a selector bar can be used to activate/disactivate different kinds of data (PINs for IoT sensors, POIs, and bus stops, multiple heatmaps, traffic information as heatmap or 3D crests/arrows, cycling paths, 3D columns, and What-If analysis). Each icon provides the specific data name by mouseover, and for some of the available data a panel is shown in the top-right corner of the main map where additional information and controls are provided for the specific layer. In the main portion of the dashboard a 3D multidata map is presented. Into the map, on the top-left corner, an in-map menu is provided with buttons to control rotation, tilt, and zoom movements. Additionally, the cross-like button can be used to put the map at full screen, while arrow and hand button can be used to toggle the building picking modality. Finally, the compass button can be used to orientate the map with the north direction pointing upward. By clicking on the gear icon, a sub-menu is shown with the following fields: Orthomaps, that can be used to change the orthomap displayed over the terrain, Buildings, to select which kind of building has to be rendered in the map, and Settings, where the user can specify different parameters to optimize his/her user experience according to the client hardware and to activate/disactivate specific data.

The Settings panel provides the user with the possibility to set the following parameters:

- Max number of tiles: this number indicates the maximum number of building tiles to be displayed in the map. By increasing the value more buildings are shown, requiring higher CPU and GPU loads. Conversely, when reducing the value, less buildings are rendered, easing the SCDT usage with low-end clients.
- Min 3D zoom: the minimum zoom value for displaying 3D buildings. If the zoom value is less than such value, no buildings are rendered into the map.
- Traffic Animation Enabled: if checked, 3D animated arrows are used to display traffic information. Otherwise, less resource consuming static 3D crests are used.
- Arrow size (m): this value specifies 3D arrow dimensions. This value indirectly impacts on the apparent arrow animation speed.
- Height crest layer (m): this value specifies the height of 3D arrows and 3D crests used to represent the traffic.
- Display road information: if checked, the road graph is requested from the KM4City knowledge base and displayed over the terrain.
- Display decorations: if checked, additional 3D elements (trees and airplanes at the airport) are shown in the map. The Min 3D zoom impacts also on this 3D elements.
- Realistic decoration: if checked, more realistic 3D trees models (requiring additional resources) are shown in the map. This setting works only if the Display decoration is active.

After setting appropriate values, by clicking the Save button, changes are applied into the map without requiring any page reload.

In the bottom part of the dashboard two additional widgets are included. On the left side, the Select a City/Location widget can be used to navigate in different locations worldwide by clicking on the predefined cities or by inserting a specific location. This is particularly useful when using Google 3D tiles having a wide coverage. On the right side, a timeseries widget is included. This widget can be used to show time evolution of specific measurements of a selected IoT device: after activating a device kind by means of a button in the selector menu, the user can click on a PIN, select the

RT DATA tab in the shown popup, and then click on one of the options under the Buttons column (i.e., 4h, 24h, 7d, 30d, 6m, 1y). Such an action provokes a client-side business logic callback that is caught by the widget that will show the timeseries for the requested period.

ACKNOWLEDGMENT

The authors would like to thank the MUR, the University of Florence, and the companies involved for co-funding the National Center on Sustainable Mobility, MOST (https://www.centronazionalemost.it/). They also thank many developers working on the Snap4City platforms. Snap4City (https://www.snap4city.org) and KM4City are open technologies of the DISIT Laboratory.

REFERENCES

[1] M. Grieves, "PLM initiatives [PowerPoint slides]," Presented at the Product Lifecycle Manag. Special Meeting, Ann Arbor, MI, USA: Lurie Engineering Center, University of Michigan, 2002.

[2] M. Grieves, "Virtually intelligent product systems: Digital and physical twins," in *Complex Systems Engineering: Theory and Practice*, S. Flumerfelt, Ed. Reston, VA, USA: American Institute of Aeronautics and Astronautics, 2019, pp. 175–200.

[3] M. W. Grieves, "Product lifecycle management: The new paradigm for enterprises," *Int. J. Product Develop.*, vol. 2, nos. 1–2, p. 71, 2005, doi: 10.1504/ijpd.2005.006669.

[4] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," in *Transdisciplinary Perspectives on Complex Systems*, J. Kahlen, S. Flumerfelt, and A. Alves, Eds. Cham, Switzerland: Springer, 2017, doi: 10.1007/978-3-319-38756-7_4.

[5] E. Glaessgen and D. Stargel, "The digital twin paradigm for future NASA and U.S. air force vehicles," in *Proc. 53rd AIAA/ASME/ASCE/AHS/ASC Struct., Struct. Dyn. Mater. Conf. 20th AIAA/ASME/AHS Adapt. Struct. Conf. 14th AIAA*, Apr. 2012, p. 1818, doi: 10.2514/6.2012-1818.

[6] F. Tao and M. Zhang, "Digital twin shop-floor: A new shop-floor paradigm towards smart manufacturing," *IEEE Access*, vol. 5, pp. 20418–20427, 2017, doi: 10.1109/ACCESS.2017.2756069.

[7] C. Cimino, E. Negri, and L. Fumagalli, "Review of digital twin applications in manufacturing," *Comput. Ind.*, vol. 113, Dec. 2019, Art. no. 103130, doi: 10.1016/j.compind.2019.103130.

[8] D.-G.-J. Opoku, S. Perera, R. Osei-Kyei, and M. Rashidi, "Digital twin application in the construction industry: A literature review," *J. Building Eng.*, vol. 40, Aug. 2021, Art. no. 102726, doi: 10.1016/j.jobe.2021.102726.

[9] A. Kumari, S. Tanwar, S. Tyagi, N. Kumar, M. Maasberg, and K.-K.-R. Choo, "Multimedia big data computing and Internet of Things applications: A taxonomy and process model," *J. Netw. Comput. Appl.*, vol. 124, pp. 169–195, Dec. 2018, doi: 10.1016/j.jnca.2018.09.014.

[10] S. Sagiroglu and D. Sinanc, "Big data: A review," in *Proc. Int. Conf. Collaboration Technol. Syst. (CTS)*, May 2013, pp. 42–47, doi: 10.1109/CTS.2013.6567202.

[11] M. Ghobakhloo, M. Fathi, M. Iranmanesh, P. Maroufkhani, and M. E. Morales, "Industry 4.0 ten years on: A bibliometric and systematic review of concepts, sustainability value drivers, and success determinants," *J. Cleaner Prod.*, vol. 302, Jun. 2021, Art. no. 127052, doi: 10.1016/j.jclepro.2021.127052.

[12] N. Elgendy and A. Elragal, "Big data analytics: A literature review paper," in *Advances in Data Mining. Applications and Theoretical Aspects ICDM* (Lecture Notes in Computer Science), vol. 8557, P. Perner, Ed. Cham, Switzerland: Springer, 2014, doi: 10.1007/978-3-319-08976-8_16.

[13] B. Ketzler, V. Naserentin, F. Latino, C. Zangelidis, L. Thuvander, and A. Logg, "Digital twins for cities: A state of the art review," *Built Environ.*, vol. 46, no. 4, pp. 547–573, Dec. 2020, doi: 10.2148/benv.46.4.547.

[14] A. Camero and E. Alba, "Smart city and information technology: A review," *Cities*, vol. 93, pp. 84–94, Oct. 2019, doi: 10.1016/j.cities.2019.04.014.

[15] European Commission. *The Future of Cities—Opportunities, Challenges and the Way Forward*. Accessed: Mar. 14, 2024. [Online]. Available: https://joint-research-centre.ec.europa.eu/jrc-news-and-updates/future-cities-opportunities-challenges-and-way-forward-2019-10-11_en

[16] United Nations. *68% of the World Population Projected to Live in Urban Areas By 2050, Says UN*. Accessed: Mar. 14, 2024. [Online]. Available: https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html

[17] European Commission. *Developments and Forecasts on Continuing Urbanisation*. Accessed: Mar. 14, 2024. [Online]. Available: https://knowledge4policy.ec.europa.eu/foresight/topic/continuing-urbanisation/developments-and-forecasts-on-continuing-urbanisation_en

[18] M. Wang and N. Debbage, "Urban morphology and traffic congestion: Longitudinal evidence from U.S. cities," *Comput., Environ. Urban Syst.*, vol. 89, Sep. 2021, Art. no. 101676, doi: 10.1016/j.compenvurbsys.2021.101676.

[19] R. P. Pradhan, M. B. Arvin, and M. Nair, "Urbanization, transportation infrastructure, ICT, and economic growth: A temporal causal analysis," *Cities*, vol. 115, Aug. 2021, Art. no. 103213, doi: 10.1016/j.cities.2021.103213.

[20] X. Zhang, L. Han, H. Wei, X. Tan, W. Zhou, W. Li, and Y. Qian, "Linking urbanization and air quality together: A review and a perspective on the future sustainable urban development," *J. Cleaner Prod.*, vol. 346, Apr. 2022, Art. no. 130988, doi: 10.1016/j.jclepro.2022.130988.

[21] Y. Yang, J. Liu, Y. Lin, and Q. Li, "The impact of urbanization on China's residential energy consumption," *Structural Change Econ. Dyn.*, vol. 49, pp. 170–182, Jun. 2019, doi: 10.1016/j.strueco.2018.09.002.

[22] L. J. Ramirez Lopez and A. I. Grijalba Castro, "Sustainability and resilience in smart city planning: A review," *Sustainability*, vol. 13, no. 1, p. 181, Dec. 2020, doi: 10.3390/su13010181.

[23] F. Biljecki, J. Stoter, H. Ledoux, S. Zlatanova, and A. Çöltekin, "Applications of 3D city models: State of the art review," *ISPRS Int. J. Geo-Information*, vol. 4, no. 4, pp. 2842–2889, Dec. 2015, doi: 10.3390/ijgi4042842.

[24] G. Gröger and L. Plümer, "CityGML–interoperable semantic 3D city models," *ISPRS J. Photogramm. Remote Sens.*, vol. 71, pp. 12–33, Jul. 2012, doi: 10.1016/j.isprsjprs.2012.04.004.

[25] E. Shahat, C. T. Hyun, and C. Yeom, "City digital twin potentials: A review and research agenda," *Sustainability*, vol. 13, no. 6, p. 3386, Mar. 2021, doi: 10.3390/su13063386.

[26] B. Lei, P. Janssen, J. Stoter, and F. Biljecki, "Challenges of urban digital twins: A systematic review and a delphi expert survey," *Autom. Construction*, vol. 147, Mar. 2023, Art. no. 104716, doi: 10.1016/j.autcon.2022.104716.

[27] L. Adreani, P. Bellini, M. Fanfani, P. Nesi, and G. Pantaleo, "Design and develop of a smart city digital twin with 3D representation and user interface for what-if analysis," in *Proc. Int. Conf. Comput. Sci. Appl.*, 2023, pp. 531–548, doi: 10.1007/978-3-031-37126-4_34.

[28] DISIT Lab. *Snap4City*. Accessed: Mar. 14, 2024. [Online]. Available: https://www.snap4city.org

[29] F. Alberti, A. Alessandrini, D. Bubboloni, C. Catalano, M. Fanfani, M. Loda, A. Marino, A. Masiero, M. Meocci, P. Nesi, and A. Paliotto, "Mobile mapping to support an integrated transport-territory modelling approach," *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 48, pp. 1–7, May 2023, doi: 10.5194/isprs-archives-XLVIII-1-W1-2023-1-2023.

[30] L. Deren, Y. Wenbo, and S. Zhenfeng, "Smart city based on digital twins," *Comput. Urban Sci.*, vol. 1, no. 1, p. 4, Dec. 2021, doi: 10.1007/s43762-021-00005-y.

[31] M. Charitonidou, "Urban scale digital twins in data-driven society: Challenging digital universalism in urban planning decision-making," *Int. J. Architectural Comput.*, vol. 20, no. 2, pp. 238–253, Jun. 2022, doi: 10.1177/14780771211070005.

[32] T. Ruohomäki, E. Airaksinen, P. Huuska, O. Kesäniemi, M. Martikka, and J. Suomisto, "Smart city platform enabling digital twin," in *Proc. Int. Conf. Intell. Syst. (IS)*, Sep. 2018, pp. 155–161, doi: 10.1109/IS.2018.8710517.

[33] M. Bauer, F. Cirillo, J. Fürst, G. Solmaz, and E. Kovacs, "Urban digital twins—A FIWARE-based model," *Automatisierungstechnik*, vol. 69, no. 12, pp. 1106–1115, Dec. 2021, doi: 10.1515/auto-2021-0083.

[34] D. N. Ford and C. M. Wolf, "Smart cities with digital twin systems for disaster management," *J. Manage. Eng.*, vol. 36, 2020, Art. no. 04020027. [Online]. Available: https://api.semanticscholar.org/CorpusID:216512763

[35] J. Yan, S. Zlatanova, M. Aleksandrov, A. A. Diakite, and C. Pettit, "Integration of 3D objects and terrain for 3D modelling supporting the digital twin," *ISPRS Ann. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. IV-4/W8, pp. 147–154, Sep. 2019, doi: 10.5194/isprs-annals-iv-4-w8-147-2019.

[36] A. A. Diakite and S. Zlatanova, "Automatic geo-referencing of BIM in GIS environments using building footprints," *Comput., Environ. Urban Syst.*, vol. 80, Mar. 2020, Art. no. 101453, doi: 10.1016/j.compenvurbsys.2019.101453.

[37] A. Hamilton, H. Wang, A. M. Tanyer, Y. Arayici, X. Zhang, and Y. Song, "Urban information model for city planning," *J. Inf. Technol. Construction*, vol. 10, pp. 55–67, Jan. 2005.

[38] J. Ferré-Bigorra, M. Casals, and M. Gangolells, "The adoption of urban digital twins," *Cities*, vol. 131, Dec. 2022, Art. no. 103905, doi: 10.1016/j.cities.2022.103905.

[39] S. Ivanov, K. Nikolskaya, G. Radchenko, L. Sokolinsky, and M. Zymbler, "Digital twin of city: Concept overview," in *Proc. Global Smart Ind. Conf. (GloSIC)*, Nov. 2020, pp. 178–186, doi: 10.1109/GloSIC50886.2020.9267879.

[40] R. D'Hauwers, N. Walravens, and P. Ballon, "From an inside-in towards an outside-out urban digital twin: Bus. models and implementation challenges," *ISPRS Ann. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 8, pp. 25–32, Sep. 2021, doi: 10.5194/isprs-annals-viii-4-w1-2021-25-2021.

[41] Z. Allam and D. S. Jones, "Future (post-COVID) digital, smart and sustainable cities in the wake of 6G: Digital twins, immersive realities and new urban economies," *Land Use Policy*, vol. 101, Feb. 2021, Art. no. 105201, doi: 10.1016/j.landusepol.2020.105201.

[42] S. H. Nguyen and T. H. Kolbe, "Modelling changes, stakeholders and their relations in semantic 3D city models," *ISPRS Ann. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 8, pp. 137–144, Oct. 2021, doi: 10.5194/isprs-annals-viii-4-w2-2021-137-2021.

[43] J. G. Singla and K. Padia, "A novel approach for generation and visualization of virtual 3D city model using open source libraries," *J. Indian Soc. Remote Sens.*, vol. 49, no. 6, pp. 1239–1244, Jun. 2021, doi: 10.1007/s12524-020-01191-8.

[44] *Stockholm Opencities Planner*. Accessed: Mar. 14, 2024. [Online]. Available: https://eu.opencitiesplanner.bentley.com/stockholm/stockholmvaxer

[45] European Commission. (Apr. 19, 2016). *EU EGovernment Action Plan 2016-2020. Accelerating the Digital Transformation of Government*. [Online]. Available: http://ec.europa.eu/newsroom/dae/document.cfm?doc_id=15268

[46] UN-Habitat. (2016). *Urban Planning and Design Lab's: Tools for Integrated and Participatory Urban Planning*. [Online]. Available: https://unhabitat.org/urban-planning-and-design-labs-tools-for-integrated-and-participatory-urban-planning

[47] M. Hämäläinen, "Urban development with dynamic digital twins in Helsinki city," *IET Smart Cities*, vol. 3, no. 4, pp. 201–210, 2021, doi: 10.1049/smc2.12015.

[48] E. Airaksinen, M. Bergstrom, H. Heinonen, K. Kaisla, K. Lahti, J. Suomisto. (May 2, 2019). *The Kalasatama Digital Twins Project. Final Report of Kira-Dig Project*. [Online]. Available: https://www.hel.fi/static/liitteet-2019/Kaupunginkanslia/Helsinki3D_Kalasatama_Digital_Twins.pdf

[49] City of Helsinki. (Mar. 14, 2024). *Helsinki Digital Twin*. Accessed: Mar. 14, 2024. [Online]. Available: https://kartta.hel.fi/?setlanguage=en#, and https://kartta.hel.fi/3d/#/

[50] Rennes, Ville et Métropole. *Rennes En 3D*. Accessed: Mar. 14, 2024. [Online]. Available: https://rennes2030.fr/rennes-en-3d/

[51] Gemeente Rotterdam. *Rotterdam 3D*. Accessed: Mar. 14, 2024. [Online]. Available: https://www.3drotterdam.nl

[52] 3D City DB. *Berlin 3D* Accessed: Mar. 14, 2024. [Online]. Available: 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%3D, and https%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%3D, and https%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

[53] G. Schrotter and C. Hürzeler, "The digital twin of the city of Zurich for urban planning," *PFG J. Photogramm., Remote Sens. Geoinformation Sci.*, vol. 88, no. 1, pp. 99–112, Feb. 2020, doi: 10.1007/s41064-020-00092-2.

[54] F. Dembski, U. Wössner, M, Letzgus, M. Ruddat, and C. Yamu, "Urban digital twins for smart cities and citizens: The case study of Herrenberg, Germany," *Sustainability*, vol. 12, no. 6, p. 2307, 2020, doi: 10.3390/su12062307.

[55] Sidewalk Labs. *Sidewalk Toronto*. Accessed: Mar. 14, 2024. [Online]. Available: https://www.sidewalklabs.com/toronto

[56] Esri. *Boston Digital Twin*. Accessed: Mar. 14, 2024. [Online]. Available: https://boston.maps.arcgis.com/apps/webappviewer3d/index.html?id=cf3415dea19d480caa71eb5dbdce185f

[57] Singapore Land Authority. (2014). *Virtual Singapore*. Accessed: Mar. 14, 2024. [Online]. Available: https://www.sla.gov.sg/articles/press-releases/2014/virtual-singapore-a-3d-city-model-platform-for-knowledge-sharing-and-community-collaboration

[58] D. Weir-McCall. *51World Creates Digital Twin of the Entire City of Shanghai*. Accessed: Mar. 14, 2024. [Online]. Available: https://www.unrealengine.com/en-U.S./spotlights/51world-creates-digital-twin-of-the-entire-city-of-shanghai

[59] The Boundary. *Wellington Digital Twin*. Accessed: Mar. 14, 2024. [Online]. Available: https://www.the-boundary.com/work/wellington-digital-twin

[60] Epic Games. *Unreal Engine Digital Twins for Architecture, Real Estate, and the Built Environment*. Accessed: Mar. 14, 2024. [Online]. Available: https://www.unrealengine.com/en-U.S./digital-twins

[61] Q. Han, P. Nesi, G. Pantaleo, and I. Paoli, "Smart city dashboards: Design, development, and evaluation," in *Proc. IEEE Int. Conf. Human-Machine Syst. (ICHMS)*, Sep. 2020, pp. 1–4, doi: 10.1109/ICHMS49158.2020.9209493.

[62] C. Garau, P. Nesi, I. Paoli, M. Paolucci, and P. Zamperlin, "A big data platform for smart and sustainable cities: Environmental monitoring case studies in Europe," in *Proc. Int. Conf. Comput. Sci. Appl. (ICCSA)*, 2020, pp. 1–12, doi: 10.1007/978-3-030-58820-5.

[63] P. Bellini, L. A. Ipsaro Palesi, A. Giovannoni, and P. Nesi, "Managing complexity of data models and performance in broker-based Internet/Web of Things architectures," *Internet Things*, vol. 23, Oct. 2023, Art. no. 100834, doi: 10.1016/j.iot.2023.100834.

[64] C. Badii, P. Bellini, A. Difino, P. Nesi, G. Pantaleo, and M. Paolucci, "MicroServices suite for smart city applications," *Sensors*, vol. 19, no. 21, p. 4798, 2019, doi: 10.3390/s19214798.

[65] A. Arman, P. Bellini, and P. Nesi, "Searching for heterogeneous GeoLocated services via API federation," in *Proc. 22nd Int. Conf. Comput. Sci. Appl. (ICCSA)*, 2022, pp. 173–190, doi: 10.1007/978-3-031-10592-0.

[66] C. Badii, P. Bellini, D. Cenni, A. Difino, P. Nesi, and M. Paolucci, "Analysis and assessment of a knowledge based smart city architecture providing service Apis," *Future Gener. Comput. Syst.*, vol. 75, pp. 14–29, Oct. 2017, doi: 10.1016/j.future.2017.05.001.

[67] F. Cirillo, G. Solmaz, E. L. Berz, M. Bauer, B. Cheng, and E. Kovacs, "A standard-based open source IoT platform: FIWARE," *IEEE Internet Things Mag.*, vol. 2, no. 3, pp. 12–18, Sep. 2019, doi: 10.1109/IOTM.0001.1800022.

[68] S. Bilotta and P. Nesi, "Estimating CO_2 emissions from IoT traffic flow sensors and reconstruction," *Sensors*, vol. 22, no. 9, p. 3382, 2022, doi: 10.3390/s22093382.

[69] S. Bilotta and P. Nesi, "Traffic flow reconstruction by solving indeterminacy on traffic distribution at junctions," *Future Gener. Comput. Syst.*, vol. 114, pp. 649–660, Jan. 2021, doi: 10.1016/j.future.2020.08.017.

[70] N. Girard, G. Charpiat, and Y. Tarabalka, "Aligning and updating cadaster maps with aerial images by multi-task, multi-resolution deep learning," in *Proc. Asian Conf. Comput. Vis. (ACCV)*, 2018, pp. 1–15, doi: 10.1007/978-3-030-20873-8.

[71] L. Adreani, P. Bellini, C. Colombo, M. Fanfani, P. Nesi, G. Pantaleo, and R. Pisanu, "Digital twin framework for smart city solutions," in *Proc. Int. Conferences Distrib. Multimedia Syst.*, Jun. 2022, pp. 1–8, doi: 10.18293/dmsviva2022-012.

[72] L. Adreani, C. Colombo, M. Fanfani, P. Nesi, G. Pantaleo, and R. Pisanu, "A photorealistic 3D city modeling framework for smart city digital twin," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Jun. 2022, pp. 299–304, doi: 10.1109/SMARTCOMP55677.2022.00071.

[73] N. R. Pal and S. K. Pal, "A review on image segmentation techniques," *Pattern Recognit.*, vol. 26, no. 9, pp. 1277–1294, Sep. 1993, doi: 10.1016/0031-3203(93)90135-j.

[74] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Advances in Knowledge Discovery and Data Mining* (Lecture Notes in Computer Science), vol. 7819, J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds. Berlin, Germany: Springer, 2013, doi: 10.1007/978-3-642-37456-2_14.

[75] R. Toldo and A. Fusiello, "Robust multiple structures estimation with J-linkage," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2008, pp. 537–547, doi: 10.1007/978-3-540-88682-2.

[76] L. Adreani, P. Bellini, C. Colombo, M. Fanfani, P. Nesi, G. Pantaleo, and R. Pisanu, "Implementing integrated digital twin modelling and representation into the Snap4City platform for smart city solutions," *Multimedia Tools Appl.*, vol. 83, no. 12, pp. 37121–37146, Oct. 2023, doi: 10.1007/s11042-023-16838-0.

[77] DISIT Lab. *3D Building Modelling: Source Code*. Accessed: Mar. 14, 2024. [Online]. Available: https://github.com/disit/3d-building-modelling

[78] Open Geospatial Consortium. *OGC 3D Tiles*. Accessed: Mar. 14, 2024. [Online]. Available: https://www.ogc.org/standard/3dtiles/

[79] P. Bellini, D. Bologna, M. Fanfani, L. A. Ipsaro Palesi, P. Nesi, and G. Pantaleo, "Rapid prototyping & development life cycle for smart applications of Internet of Entities," in *Proc. 27th Int. Conf. Eng. Complex Comput. Syst. (ICECCS)*, Jun. 2023, doi: 10.1109/iceccs59891.2023.00026.

[80] Astronomy Answers. *Position of the Sun*. Accessed: Mar. 14, 2024. [Online]. Available: https://www.aa.quae.nl/en/reken/zonpositie.html

[81] P. Bellini, S. Bilotta, A. L. I. Palesi, P. Nesi, and G. Pantaleo, "Vehicular traffic flow reconstruction analysis to mitigate scenarios with large city changes," *IEEE Access*, vol. 10, pp. 131061–131075, 2022, doi: 10.1109/ACCESS.2022.3229183.

[82] DISIT Lab. *Client-Side Business Logic*. Accessed: Mar. 14, 2024. [Online]. Available: https://www.snap4city.org/download/video/ClientSideBusinessLogic-WidgetManual.pdf

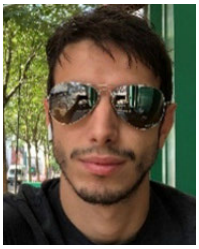
[83] P. Nesi, L. Po, J. R. R Viqueira, and R. Trillo-Lado, "An integrated smart city platform," in *Semantic Keyword-Based Search on Structured Data Sources* (Lecture Notes in Computer Science), vol. 10546, J. Szymański and Y. Velegrakis, Eds. Cham, Switzerland: Springer, 2017, doi: 10.1007/978-3-319-74497-1_17.

[84] P. Bellini, F. Bugli, P. Nesi, G. Pantaleo, M. Paolucci, and I. Zaza, "Data flow management and visual analytic for big data smart City/IoT," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Aug. 2019, doi: 10.1109/smartworld-uic-atc-scalcom-iop-sci.2019.00276.

[85] DISIT Lab. *Containerized Version of Snap4City Based on Docker Compose*. Accessed: Mar. 14, 2024. [Online]. Available: https://www.snap4city.org/docker-generator/selecting_model

[86] OpenStreetMap. *OSM Buildings*. Accessed: Mar. 14, 2024. [Online]. Available: https://osmbuildings.org

[87] Google. *Google 3D Tiles*. Accessed: Mar. 14, 2024. [Online]. Available: https://developers.google.com/maps/documentation/tile/3d-tiles



LORENZO ADREANI is currently pursuing the master's degree in information engineering with the University of Florence. He is with the DISIT Laboratory, University of Florence. His research interests include digital twins, computer graphics, and knowledge engineering.

PIERFRANCESCO BELLINI received the degree in computer engineering and the Ph.D. degree in computer engineering and telecommunications from the University of Florence. He is currently a Professor and a Researcher with the Department of Information Engineering, University of Florence, and a Professor of operating systems. His main research interests include semantic computing, ontology engineering, and cloud computing.

MARCO FANFANI received the Ph.D. degree in computer engineering, multimedia, and telecommunications from the University of Florence. He is currently a Research Fellow with the Department of Information Engineering, University of Florence. He worked on several national and international research projects. His research interests include digital twins, the IoT, knowledge engineering, AI, computer vision, and 3D reconstruction.

PAOLO NESI (Member, IEEE) is currently a Full Professor with the DINFO, University of Florence, and the Chief of the DISIT Laboratory and Snap4City. He is and has been the coordinator of several multi-partner international research and development projects. He has published more than 400 papers on international journals and conferences. His research interests include machine learning, massive parallel and distributed systems, physical models, the IoT, mobility, big data analytic, AI/XAI, semantic computing, formal model, machine learning, and data privacy. He has been the chair of a number of international conferences of IEEE and other organizations.

GIANNI PANTALEO is currently an Aggregated Professor with the DINFO, University of Florence. He has been the coordinator of a number of WPs in international research and development projects. His research interests include knowledge engineering, the IoT, visual analytics, mobility, NLP, semantic computing, and visual analytics.

Open Access funding provided by 'Università degli Studi di Firenze' within the CRUI CARE Agreement