A Photorealistic 3D City Modeling Framework for Smart City Digital Twin

Lorenzo Adreani[1], Carlo Colombo[2], Marco Fanfani[1,2], Paolo Nesi[1], Gianni Pantaleo[1], Riccardo Pisanu[2]
University of Florence, Florence, Italy, email: <name>.<surname>@unifi.it
1) DISIT lab, https://www.disit.org, https://www.snap4city.org
2) Computational Vision Group http://cvg.dsi.unifi.it/cvg/

***Abstract*— In the context of Smart Cities digital transformation, the field of 3D city modelling has attracted a growing interest for representing the city digital twin. This paper presents a method for producing a 3D city model with photorealistic rooftop textures extracted from aerial images, as well as the integration of the 3D city model into an open-source Smart City framework. The proposed solution provides a smart visualization of 3D city entities integrated with a large variety of Smart City data (coming, for instance, from IoT Devices which generate time-series data, heatmaps, geometries and shapes related to traffic flows, bus routes, cycling paths etc.). The proposed method for rooftop detection and alignment follows a deep learning approach based on U-Net architecture, and it has been validated against a manually created ground-truth of 50 buildings scattered uniformly on the covered area. The solution is implemented in the open-source Snap4City Smart City platform.**

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

I. INTRODUCTION

In smart city solutions, spatial data information may act as a key driver in order to enhance the interoperability among all these systems [1]. Aspects related to digital 3D city modelling and digital twin have recently gained a growing interest, since they allow to perform analyses, simulations, planning and monitoring in several different domains and application areas, thus improving decision-making processes for Smart Cities stakeholders. Many approaches have been proposed in literature, such as: CityGML, CityJSON, the combination of Building Information Modeling (BIM) and Geographic Information System (GIS) providing a City Information Modeling (CIM) [2]. CityGML also defines different levels of detail (LoD) for the models. According to [3], there are five levels of detail: LoD0 is represented by those models having only a 2D map with 3D terrain; LoD1 presents buildings as simple boxes; LoD2 adds rooftops details to LoD1 buildings; LoD3 presents also external facades structure; LoD4 adds building interiors. LoD4 was introduced in CityGML 2.0, but it was removed in the latest version of CityGML 3.0. Generally, performances and scalability of these systems are some challenging aspects, due to the large amount of data to be processed. This impacts especially when a higher level of detail is provided, for instance when applying photorealistic textures. In these cases, the issue is typically mitigated at the expense of a lower resolution of textures.

In this paper, a 3D City Modeling Framework for Smart City Digital Twin with photorealistic textures is presented. The main contributions of this paper are the following: first, the production of a full functional 3D map with LoD3 model support (providing photorealistic rooftop textures), as well as the creation of a full automatized algorithm to map all the buildings in a certain area, creating a LoD2 building models from a LoD1 building type. Second, the integration of the 3D map into a Smart City framework (the open-source Snap4City platform) [4], in order to provide a smart environment and applications for visualizing city entities and related data (coming, for instance, from IoT devices generating time-series data, heatmaps, geometries and shapes related to traffic flows, bus routes, cycling paths etc.), with the possibility to pick single city elements or buildings on map, and inspect their data and attributes. Therefore, the proposed solution is an open-source web-based tool for producing a global digital twin integrating IoT and many other kind of Smart City data, which has been designed to satisfy the following requirements:

1) *Production of a photorealistic 3D City model as an interactive map*: the map must support a full 3D environment, implementing a 3D city model with photorealistic textures for building rooftops. The map must be interactive, allowing the user to click on single city elements, device markers and buildings, and inspect all the attached data and attributes, as detailed in requirement #2.
2) *The map must support the retrieval and visualization of the many different data sources collected in the Snap4City Smart City platform* (as described in Section IV), for instance: IoT devices, Points of Interests (POI), heatmaps, buildings information, geometries and polylines related to bus routes, cycling paths, traffic flows etc. The IoT devices typically have time series data attached to represent the evolution of any aspect of the digital twin, e.g.: temperature, pressure, speed, vibration, water level, pollutant concentration, number of people visiting, etc.
3) *LoD1, LoD2, LoD3 buildings support*: All levels of LoD detail must be implemented, except LoD4 buildings.
4) *WMS support*: the Web Map Service (WMS) protocol must be integrated in the application.
5) *Terrain Elevation*: the implementation and management of 3D Terrain via Digital Terrain Model (DTM) heatmap must be supported.
6) *Customizable Orthomap*: The base orthomap can be changed at runtime, selecting from a pool of available orthomaps.

The rest of the paper is organized as follows: in Section II, a review of the state of the art is presented, describing the works

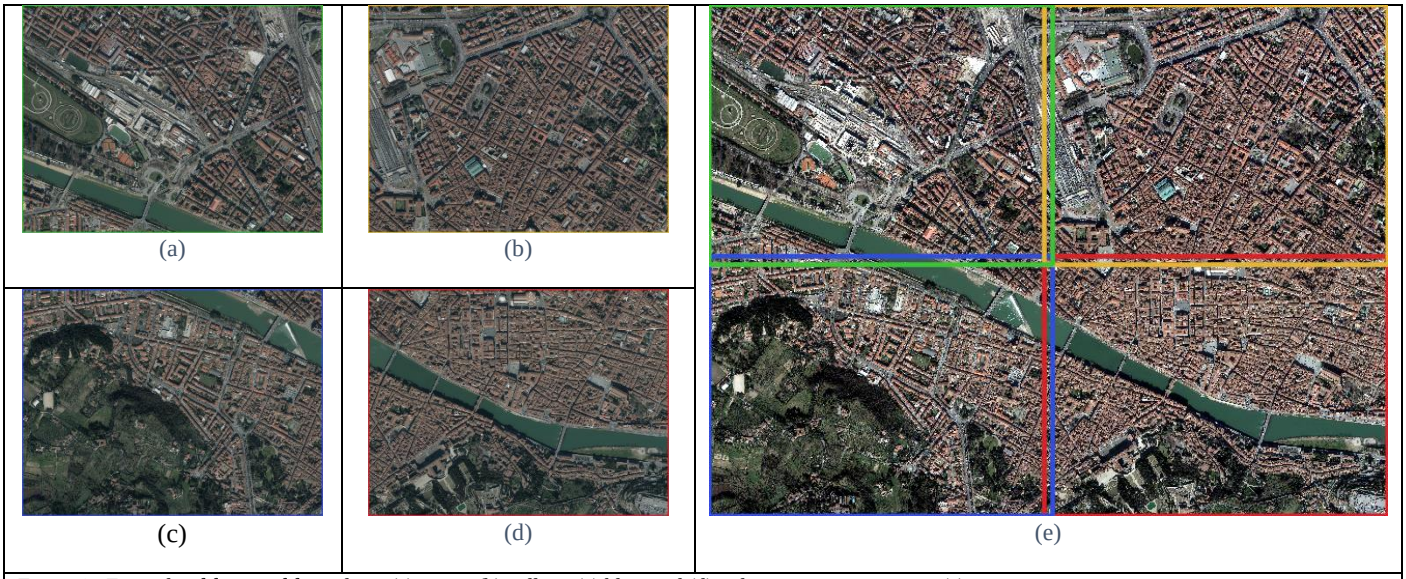

Figure 1: Example of fusion of four tiles – (a) green, (b) yellow, (c) blue, and (d) red – into a unique image (e).

in literature related to 3D city models and extraction and registration of photorealistic textures; Section III describes the proposed method for producing the photorealistic textured 3D map; Section IV presents the integration of the proposed method in the Snap4City platform; in Section V, a quantitative validation of rooftop alignment is presented; finally, Section VI is left for conclusions.

II. Related Works

In the past years a lot of research has been made in the field of 3D city modelling, trying to recreate a realistic visualization based on digital data. However, due to the typical size of a city, handling all the data and their processing is a challenging task, thus several challenges remain unresolved yet [5]. One critical aspect of developing a high-fidelity 3D city model is to find the correct format for the data that will be sent to the application. The CityGML and CityJSON standards have been mainly adopted in the past years: they define a format for the representation of geometry and topology for 3D buildings, using respectively XML and JSON. CityGML 3.0 integrates also the BIM standard, alongside the GIS format, from Industrial foundation class (IFC) [6]. Some integrations of CityGML have been proposed in real cases, such as the city of Helsinki, in which a LoD3 city model was implemented and made publicly available [7]; however, the system do not provide integration with IoT or other kind of city data. An attempt of making a LoD3 3D city model was made by ETH Zurich with the VarCity project [8]. However, the provided semantic information is generally limited to a small number of semantic classes.

Advancements in Light Detection And Ranging (LiDAR) technology allow to model urban topography at spatial resolution and granularity which were not achievable before the advent of this technology [9]. In [10], a method to create a city model from a point cloud generated by LiDAR technology is presented. This approach has shown to reduce the time to generate the model, but it cannot process unsymmetrical objects and present some geometrical error.

In order to enhance a 3D map with photorealistic textures of the rooftops of the city buildings, two main aspects must be considered: at first, rooftops have to be detected in remote sensed data (i.e., airborne or satellite images) [11]; then, the segmented patches must be carefully aligned with the top-view of the 3D map. Indeed, even if geolocalization information is available, errors are present due to uncertainties [12] and an accurate multi-modal registration (e.g., between the RGB images and the 3D structure) is required [13].

In the literature, several works have addressed these topics using both standard and learning-based solutions. In [14], handcrafted features and a hierarchical segmentation approach have been used to identify the buildings in rural areas. SVMs [15] and Random Forests [16] have also been used to address this task. For example, in [17] the authors propose a three-steps method based on color-based clustering, roof detection using an SVM and a final false negative recovery. Slightly different, in [18] a pair-wise exploitation of satellite images is used to reconstruct a 3D model that could then be employed to identify rooftop regions. However, such solutions not only have some limitations when working on areas with dense buildings, but they also require a successive registration on the 3D map.

More recently, deep learning based solutions appeared for remote sensed image processing [19], [20]. More closely related to our task, in [21] a Mask R-CNN [22] was used to detect rooftops from aerial images. Differently, in [23], [24] a U-Net architecture [25] has been preferred. Moreover, these last two solutions provide not only rooftop segmentation but also the registration on 3D data, making them particularly interesting for our purposes.

III. Proposed Method

In this section, the steps required to produce the textured 3D map to be deployed in the Snap4City platform will be described

[4], [26]. Before presenting the operative pipeline, the used data will be described in the following subsection.

A. Data

To build our textured model we used aerial orthographic photos of the city of Florence, kindly provided by the "Sistema Informativo Territoriale ed Ambientale" of Tuscany Region. These images are tiles with a resolution of 8200x6200 pixels with partial overlap and rough geolocalization in the EPSG 3003 (Monte Mario / Italy zone 1) coordinate system.

The 3D map was instead built by extruding the 2D shapes of the buildings according to their elevation obtained from GeoJSON provided by Florence municipality and Tuscany Region Open Data. These data are expressed in the EPSG 4326 (WGS84) coordinate system.

B. Operative pipeline

1) Pre-processing. At first, the airborne images and the 2D shapes used to create the 3D map must be converted into a common coordinate reference system. We noticed that by simply moving the orthographic photos from EPSG 3003 directly to EPSG 4326 produced evident alterations in the Ground Sample Distance (GSD)[1]. To avoid this effect, we instead selected a common coordinate system: the EPSG 3857 (WGS84 / Pseudo-Mercator). Both the images and the 2D shapes were projected to it, since it correctly maintains the GSD. Secondly, multiple image tiles describing the area covered by the 3D map were fused into a single image using the GDAL library[2] (see Figure 1). Finally, we down-sampled the input image: in this way we were able to obtain a notable speed-up in the successive steps of the pipeline, without losing accuracy in the detection and alignment of the rooftops.

2) Rooftop detection and alignment. To detect the rooftops from the aerial images and align them with the 3D map, we used the method presented in [24], based on a double U-Net architecture exploiting multi-resolution [27] and multi-task learning [28]. The net takes in input an aerial RGB image and a cadastral map of building (represented as a binary image) and outputs a list of multi-polygons aligned to the input RGB image.

In our setup, we used the 2D shapes described in the GeoJSON data to obtain a cadastral map of the area of interest: a binary cadastral map was computed by rasterizing the multi-polygons – extracted from the 2D shapes – describing the buildings contours. After this step a list of multi-polygons aligned to the input RGB image was obtained. So we were able to both segment rooftop textures and compute their registration, w.r.t. the 3D map.

3) Image warping. Firstly, the aligned multi-polygons were up-scaled to take into account the image down-sampling done in step 1. Then, an affine transformation to warp the image and register it w.r.t. the 3D map was computed. However, using a single transformation for all the multi-polygons give rise to local inaccuracies. For this reason, we computed a dedicated transformation for each polygon and locally warped the image so as to obtain a better registration. So, given the vertexes of an aligned polygon V_A and the vertexes of the corresponding 2D shapes V_S (consistent with the position of rooftops in the 3D map), an affine transformation T was estimated such as

$$V_S = TV_A \tag{1}$$

Then, according to the specific T, the aerial image was warped and segmented on the area of the considered rooftop of the 3D map. After repeating this process for all the multi-polygons, a compete warped image (including only the rooftops) was obtained.

Figure 2: 3D Multi Data Map of Snap4City with textures.

[1] The Ground Sample Distance is the distance between pixel centers measured on the ground (in meters), i.e., how many meters are distant two adjacent pixels in the image.

[2] https://gdal.org/

4) 3D Model texturing. The 3D map construction and texturing were carried out with *Blender*. In particular, the building 3D models were obtained by extrusion from the 2D shapes exploiting their elevation data (included in the GeoJSON) with the *BlenderGIS* library[3]. Then a UV-map of the roof areas was created by retrieving the surfaces with normal vectors perpendicular to the main plane and the warped aerial image was used to texture the polygons described in the UV-map, using the *Python Blender API*[4]. In order to achieve a more pleasing final result. Finally, the obtained 3D map was exported in glTF format (including 3D models, textures, and coordinates) ready to be deployed in the Snap4City platform.

5) Deploy in Snap4City. The obtained model was finally integrated in the 3D Multi Data Map of the Snap4City platform, representing the digital twin of the city of Florence, powered by the *deck.gl* framework[5]. To deploy the textured 3D map, the *SceneGraphLayer* was used to import and show the glTF file. Details of the integration in the Snap4City platform are discussed in the next Section.

IV. Use case: integration in Snap4City

Snap4City is an open-source platform developed at DISIT Lab, University of Florence (https://www.snap4city.org/) [26], [29], [30]. The platform manages heterogeneous data sources, such as: IoT devices (city sensors and actuators, as well as private devices, supporting a large variety of brokers and protocols), open data, external services. For each different kind of data, static attributes (such as geographical information and other metadata) and also real-time data (when available) are collected. Device data are semantically indexed in a RDF Knowledge Base, thus they can be retrieved by dedicated APIs and exploited by Data Analytics processes and IoT applications to perform analyses, simulations, forecasts etc. This allows users to produce new knowledge on data, which can be shown on user interface through Dashboards and a wide range of widgets (showing data both in pull and push modalities). The purpose of integrating the photorealistic 3D city model obtained with the method described in Section III into the Snap4City platform is to provide a Multi-Data map which can allow the visualization of an interactive 3D environment of the city, with the possibility of inspecting the different kinds of entities and related data, such as: IoT devices, Points of Interests (POI), heatmaps, geometries related to bus routes, cycle paths, traffic flows, etc. In this way, the Snap4City platform allows to exploit a complete open-source framework that can collect, process, and manage all the data needed to obtain a high-fidelity Smart City digital twin.

In order to integrate the 3D map in the Snap4City platform, the deck.gl open-source library has been used. By exploiting the multi-layer structure of deck.gl, we can implement a layer for every type of data supported by the platform. All layers can be viewed and removed dynamically by user choice. An example of the resulting 3D map is shown in Figure 2: the 3D map can be instantiated by users as a customizable widget in their own dashboards. Figure 2 represents the 3D city model with the addition of textures obtained using the method described in Section III.

Regarding the implementation in deck.gl, first an IconLayer was implemented to represent all the IoT devices managed by the platform. IoT devices are ingested and stored in a semantic Knowledge Base, and they are classified by semantic categories. Therefore, a pool with different icons for each type of device category is used to represent device markers on map. The user can access all information given by a specific sensor and city element by simply clicking on the device marker; in this way, a popup is shown presenting static attributes and, when available, real-time and historical data can be selected and viewed on dedicated time-trend and single-content widgets (as the one shown below the map in Figure 2). Additional layers, based on the deck.gl PathLayer, have been also implemented to show different type of geometries like cycling paths, traffic flows, bus routes etc.

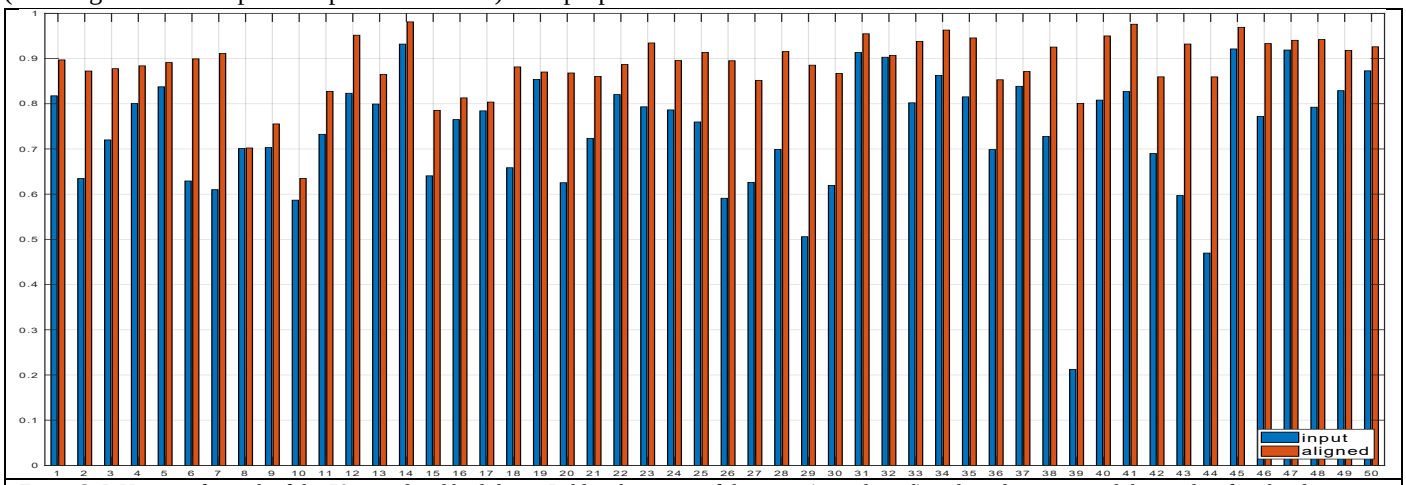

Figure 3: IoU scores for each of the 50 considered buildings. In blue the scores of the input (non-aligned) multi-polygons, in red the results after the alignment. As can be seen, IoU increase for all the buildings on the aligned multi-polygons.

[3] https://github.com/domlysz/BlenderGIS

[4] https://docs.blender.org/api/current/

[5] https://deck.gl/

In order to implement all LoD types of buildings, we support different data formats to better optimize loading time in regard of what type of LoD are requested by the user. For LoD1 type, data is stored in a GeoJSON file and loaded through a GeoJSON layer. This layer simply extrudes the base polygon to a given height (in meters). LoD2 types are implemented via the SimpleMashLayer that support the OBJ file formats, which can be shown with or without texture. Finally, to implement LoD3 type building the deck.gl SceneGraphLayer has been used, which can achieve impressive visualization without impacting too much on the application performances. This also supports two different file formats: glTF and GLB, which include texture for the building by default.

The platform also supports the visualization of heatmaps, which are essentials to provide a fast look for large amounts of data. To implement heatmap visualization in deck.gl, we used a composite layer that automatically retrieves heatmaps from a dedicated geo-server (through several formats, including WMS) and displays it as an image. Heatmaps can be static or animated; static heatmaps are viewed as single PNG images, while animated ones are sent by the geo-server in GIF format, and they are later divided in multiple images and rendered sequentially with a customizable delay time.

In addition, the elevation of terrain has been also modeled as a heatmap. In this case, the elevation mapping of a certain region has been used to elevate the base map, in order to have a 3D visualization of terrain. A TerrainLayer has been implemented for this purpose, combining the base orthomap with DTM data: the orthomap is exploited to get the ground texture, while the DTM is used for the elevation model. The result is a 3D representation of terrain with texture that better represent the territory.

V. Validation and Discussion

To obtain a quantitative validation of the rooftop alignment results on our data, we manually created a set of ground-truth multi-polygon for 50 buildings scattered uniformly on the covered area. Then we evaluate the Intersection over Union (IoU) between the ground-truth and the input (non-aligned) and aligned multi-polygons.

In Figure 3 a bar plot is reported showing the IoU score obtained for each considered building. As can be seen, for all the test cases, the IoU increases after the alignment, confirming the effectiveness of the used approach. In average we obtain an IoU score of 0.7370 for the input multi-polygons, and 0.8848 after the alignment, with an increase of almost 15%.

However, some failures are also present, and in Figure 4 we reported the most evident cases. Sometimes the input multi-polygons were far from the corresponding rooftops, and due to the density of buildings in our data, the registration failed attaching an edge of the multi-polygon on the adjacent rooftop (see Figure 4a). The alignment cannot work also in the case of occlusion (Figure 4b) or demolished buildings (Figure 4c). Finally, if the input multi-polygon does not cover all the rooftop (due to introduction of new structure), the alignment is not able to stretch the multi-polygon effectively (see Figure 4d).

VI. Conclusions

In this paper, a system for implementing a 3D city model with photorealistic texture integrated into a Smart City framework has been presented. The proposed solution follows a deep learning approach based on U-Net to detect the rooftops from aerial images and align them with the 3D map buildings,

Figure 4 - Example of failures. In red the input multi-polygons, in green the aligned ones. In (a) an edge of the multi-polygon is erroneously attached to the adjacent building. In (b) some trees occlude the view of the rooftop, while in (c) the building was demolished. Finally, in (d) the input multi-polygon does not include a new structure (on the bottom-right part of the building) and the alignment fails consequently.

which are obtained by extrusion from GeoJSON data. The solution is implemented in the open-source Snap4City platform as a multi-layer 3D map, which can be used by users as a widget on dashboards to visualize a full 3D city environment and a large variety of data, including IoT devices (city sensors and actuators, as well as private devices), POI, heatmaps, geometries and polylines related to cycling paths, bus routes, traffic flow etc. Specifically, users have the possibility to pick on map the single city elements and device markers and inspect their data and attributes. In this way, the proposed solution aims at providing an easy and smart navigation of the global digital twin of the city and the related data. The method employed for rooftop detection and alignment was validated against a set of ground-truth multi-polygon for 50 buildings, composed by uniformly scattered aerial images of the metropolitan area of Florence, achieving an IoU score of 0.7370 for the input multi-polygons, and 0.8848 after the alignment. As a future work, an automatic procedure is going to be developed, in order to apply photorealistic texture also to building facades.

ACKNOWLEDGMENT

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

REFERENCES

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[26] P. Bellini, F. Bugli, P. Nesi, G. Pantaleo, M. Paolucci, I. Zaza, "Data Flow Management and Visual Analytic for Big Data Smart City/IOT", 19th IEEE Int. Conf. on Scalable Computing and Communication, IEEE SCALCOM 2019, Leicester, UK

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

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

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

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