AN EXTENSION OF CITYJSON TO SUPPORT POINT CLOUDS

G.-A. Nys [1,*], A. Kharroubi [1], F. Poux [1], R. Billen [1]

[1] Geomatics Unit, UR SPHERES, University of Liège (ULiège), Allée du six Août, 19, 4000 Liège, Belgium - (ganys, akharroubi, fpoux, rbillen)@uliege.be

KEY WORDS: 3D City Models, CityJSON, CityGML, Smart Cities, Point Cloud

ABSTRACT:

The combination between dense point clouds and 3D vector objects permits new cartographic representation of urban information. This paper proposes an extension for the CityJSON encoding to support point clouds. Following the 3.0 CityGML specifications, attributes and features are added to the core module of v1.0.1 CityJSON. Two solutions are proposed: inline complex geometries and external link to a remote file. The extended schema can be illustrated in four scenarios: detailed features visualization, fall-back solution in features reconstruction processes, simulating urban climate represented as vector fields, and true-to-life representation solution for complex elements such as solitary vegetation objects. It permits 3D city modelers to handle points clouds in a native way reducing files size and avoiding redundancy. All developments and documentation are available open-source.

1. INTRODUCTION

3D point cloud is an emerging information representation mode. Interest in its support is growing fast. It is often the groundwork for decision-making applications used under as-built or updating conditions (monitoring, inspection, control, etc.). Improving geographical information systems to support them could open possibilities and even more so in the urban built environment. The last version of the OGC CityGML Standard v3.0 provides new features to support 3D Point Clouds in city objects and furniture. It is a significant improvement of its core module and opens new visualization and rendering capabilities (Kutzner et al., 2020).

Still, this conceptual change needs to be translated into a practical solution. Besides, the current version of CityJSON (v1.0.1) implements most of the CityGML v2.0 data model using the lightweight JSON encoding of the CityGML data model (Ledoux et al., 2019). It is the new promising way to handle geoinformation following the aforementioned CityGML data model. While benefitting the CityJSON encoding, the necessity to support 3D Point Clouds should be investigated in line with the 3.0 version of CityGML.

The paper aims to propose the translation of the new point cloud module of CityGML v3.0 into a CityJSON extension. This integrated 3D point cloud support solves several problems encountered in the last years, as illustrated in the use cases. The rest of the paper is organized as follows: in section 2, we present a state-of-the-art management of point clouds in web geographical information systems. Section 3 develops our methodology for the implementation of the extension. In section 4, we discuss and evaluate its support and illustrate the defined extension in various use cases:
(1) Fall-back solution in the case where a vector geometry is difficult to generate;
(2) High-detailed interiors for building features;
(3) Simulating urban climate represented as vector fields;
(4) True-to-life representation solution for complex elements such as solitary vegetation.

In the same section, we discuss results and draw future works for this field of research. The conclusion is presented in Section 5.

2. RELATED WORKS

Point clouds are relevent data sources in the urban built environment (Wang et al., 2020). They are considered primary sources of geometric information in many processes as the efficiency and stability of acquisition systems improve. Moreover, information on surface can be assessed, including the nature of the material thanks to the response intensity.

Even if points are basic primitives, alone, they are not very relevant: point clouds need to aggregate a huge number of them to represent elements in details. Hence, the processing and features extraction of a significant number of points can be troublesome (Wang et al., 2020). Because of the performances needs, data segmentation is a necessary step before any practical use. Semantic information linked to point batches improve the knowledge on captured factual information (Poux et al., 2017). In a second step, the abstraction of clouds and their distribution in objects classes open modelling capabilities.

A mixed visualization between dense and semantically rich 3D point clouds and abstracted cartographic objects can enjoy the benefits of both specificities (Nebiker et al., 2010). For instance, in the urban built environment, typical actors such as pedestrians and drivers can be aware of the reality on the ground thanks to this mixed representation micro-scale environment. The abstraction imposed by city modelling standards might thus be avoided (e.g., trees are often generalized as cones or spheres on the top of a cylinder).

Maintaining point clouds as they stand allows a more efficient and reliable analysis while keeping a complex rendering when it comes helpful (Brunnhuber et al., 2017). The visual expression of point cloud models reveals the physical form of the environment (Urech et al., 2020). Employing point cloud models as a source for design development in landscape projects avoids smoothing and distorting real world scenes.

* Corresponding author

Forward, it can be used to create audio-visual landscape stimuli and thus improve the immersion in a virtual environment (Spielhofer et al., 2017).

Beside the semantic usability of a mixed representation, point clouds and objects meshes are not the same from a technical and format point of view. An effort should be placed in rendering both points and objects concurrently. Once again, performances are point of interest. Some common web-related libraries are limited to efficient visualization but do not handle semantic information natively (ThreeJS and Potree based on WebGL) (Buyukdemircioglu & Kocaman, 2018). It is therefore necessary to propose another solution for the formatting of this information. Besides, rendering point clouds can be performance intensive and thus do not perform well in geographic librairies (e.g. CesiumJS). The balance between the two need to be found (Discher et al., 2019).

Recently, the new CityJSON facilitates the evolution of advance decision-making processes. For instance, its lightness and ease to maintain are demonstrated in flood simulations and complex semantic view analysis (Kumar et al., 2018; Virtanen et al., 2021). Still, its core module (v1.0.1) has not been much extended for now, with one exception on topological representation (Stelios Vitalis et al., 2019).

Besides, technical developments around CityJSON offer more and more application areas and tools for the management of 3D city models: the QGIS plugin (Stelios Vitalis et al., 2020), the automatic 3D model generation based on airborne LiDAR point cloud and the native support of metadata (Nys, Billen, et al., 2020), etc.

3. METHODOLOGY

This section states the different choices and decisions that lead to creating the point cloud extension for v1.0.1 CityJSON. The will is to propose a method that is easily reproducible and scalable due to the cumbersome and sometimes very complex nature of point clouds. The concern is to handle standard point clouds formats into a 3D model, not to process point clouds to create 3D models. Both modes are considered as complementary since they propose different information representations with their benefits.

3.1 CityJSON extensibility

The purpose of our proposition is to create an extension of CityJSON to support the integration of a PointCloud module according to the CityGML v3.0 specifications in JSON schemas. A CityJSON extension is a JSON file that documents how the core data model (i.e. JSON schemas) may be extended to increase its features and/or attributes. It differs from the CityGML ADEs, which impose software to handle additional features and thus need them to extend their support (Biljecki et al., 2018). Even if ADEs are considered as built-in mechanisms, the addition of information through ADEs impose the creation of new tables to store and encode it in relational models.

This statement is not true in the case of CityJSON if relational databases are abandoned in favour of file-based or even NoSQL management. Additional information is inserted in the initial model thanks to the JSON-encoding of the data model and its native JSON objects-based management. Figure 1 schematizes how the additional information are handled in both XML and JSON encoding of the CityGML data model.

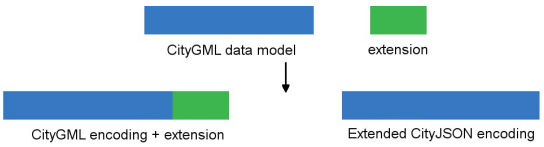

Figure 1. Extensions of the encoding

For the reminder, extending the CityGML core model's specifications could lead to software issues that are based on its XML-encoding (i.e. storage in relational databases). The vast majority of software support the addition of extensions but are limited because if relies on the XML-encoding. An additional effort is mandatory to process extended models. Conversely, extended CityJSON files remain CityJSON files and might improve the dataset information, including the integrated metadata (Nys, Poux, et al., 2020). Hence, providing an integrated and convenient extension is a great practical advance for existing CityJSON datasets and all the software supporting it. If the software does not include the extension support, it is neglected but the remaining file is still consistent and does not limit its use.

3.2 Features improvements

Extending CityJSON specifications might improve three different methods: creating a new object class, extending a model's properties, or extending the feature's attribute. While the two former solution are not relevant in this case, the latter is the most beneficial. The CityGML PointCloud module is presented as an improvement to all *CityObjects*. The model is not altered itself and the addition of *CityObjects* does not need to be investigated.

Following the CityGML 3.0 specifications, the point cloud module should support the storage of information related to point clouds differently: either inline directly within the model in a geometric aggregate such as MultiPoint, either as a link to external resource (i.e. common type such as LAS file). The usage of absolute URIs (Uniform Resource Identifier) should be preferred. Note that, since the CityJSON specifications do not yet support all the 3.0 CityGML data model features, the extension has been modelled at a high level in the features modelling. Hence, thanks to the features' hierarchy, the notion of space is neglected for the moment but will easily be updated once a new version will be available.

For the former solution (i.e. the inline geometry), some features type already support a representation mode as MultiPoint geometries (i.e. *Installations*, *SolitaryVegetationObjects*, etc.). In the case where it was not yet possible, a change in the supported object geometries for each object type is made. This modification is straightforward and does not require any other update of the JSON schema except the addition of the MultiPoint value in the enumerated supported geometry types:

```
"_AbstractBuilding": {
   "geometry": {
      "type": "array",
      "items": {
         "oneOf":          [{          "+$ref":
"../geomptrimitives.schema.json#/MultiPoint" }]
      }
   }
}
```

For the second, a web address (i.e. URI) provides a direct download link of an external resource. While the resource can be referred using a local path, an absolute path could take advantage of the availability of open datasets. One can imagine that registered clouds from cultural heritage elements or simple building greatly improve the rendering of a scene. No limit or characteristic are specified for the resource apart its format.

Indeed, some information on the resource are mandatory but does not limit the number of points, the file size, etc.: its *MimeType* attribute, the two-parts label used to identify a type of data transmitted on the web; the *pointFile* attribute, the external link itself; and the *pointFileSrName*, the identifier of the spatial reference system in which the cloud has been registered (if available). The CityJSON extension proposes to handle this information in a simple JSON object *pointcloud-file*. This new JSON object can be nested in every CityObject if needed:

```
"+pointcloud-file": {
   "type": "object",
   "properties": {
      "mimeType": {
         "type": "string"
      },
      "pointFile": {
         "type": "string",
         "format": "uri-reference"
      },
      "pointFileSrsName": {
         "type": "string",
         "default": "EPSG:4326"
      }
   }
}
```

The rendering of both representation mode formatted in the CityJSON extension has been made in a WebGL environment based on the NINJA viewer (S. Vitalis et al., 2020). The support of point's geometries needed a small improvement of the software. However, the ease of extending the CityJSON capabilities has been demonstrated in the software component update. The code is open source and available on the web under an Apache-2.0 license (https://github.com/GANys/Measur3D). There is no difference with other objects: point clouds can be selected, their attributes can be update or deleted, some can be added, etc.

4. EXPERIMENT RESULTS AND DISCUSSION

This section discusses the application of the defined extension in various uses cases and examples. Both inline and external link solutions are illustrated in the same example. Its documentation and the updated CityJSON schemas, some example can be found in the open-source git repository of the extension project on the web (https://github.com/GANys/cityjson-pointcloud).

4.1 Object generation fall-back solution

No limitation to using the MultiPoint geometry as city features representation has been formulated in the specifications. Hence, an airborne point cloud can itself be a good city model portrayal if it complies with minimal conditions. Even though it may seem like an aberration, a point cloud, which is correctly segmented and classified, is thus a valid CityJSON model. For instance, the LAS classification of airborne LiDAR data can be used in order to decompose the point cloud into different object types. A generic mapping can link a batch of points to the semantic definition of a city object type.

Afterwards, the city objects are enriched by semantic information and metadata. It is here important to note that this link can be done at the level of the object and not on the whole point cloud. One can improve batches independently and perhaps follow its evolution through the city model versions. Furthermore, an independent management of the objects and their geometries allows handling object attributes without mobilizing resources for the geometry (geometries are the heavy part of objects). This object-based management provides also all the CRUD operations (Create, Read, Update, Delete) even if features are initially part of a more general point clouds. The Figure 2 gives an extreme example where the entire model is represented as the aggregation of a point sub-cloud for every city object.

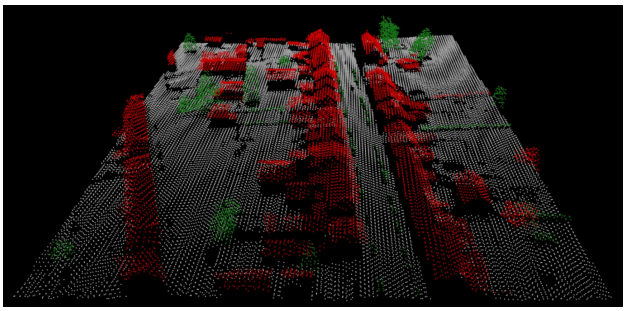

Figure 2. Visualization of an urban model CityJSON in point cloud only (33.811 points)

4.2 High-detailed features

In a more hybrid way, illustrations of combined geometry types solutions are also possible: we used such added modelling support in particular when a roof reconstruction process did not find any consistent solution for the roof shapes (Nys, Poux, et al., 2020). The point cloud offers a satisfactory fall-back result for the representation of the complexity of the roofs. If the roof shape has not been generalized, it can be rendered as points, a mesh, etc.

Furthermore, some interesting features can be extracted from clouds and meshes. Even if planes and 3D objects have not been correctly generated, some usage can still be performed. It does not limit operations such as normals evaluations. The normals are for instance useful to determine the orientation and the slope of urban roofs. These characteristics are mandatory to determine the potential of urban greening with green roofs. Urban green infrastructures impose a slope lesser than 5° and a minimum area of 10 square meters (the area can be obtained by projecting points and meshes on the horizontal plane and determining their minimal bounding box to within a factor of the slope) (Joshi et al., 2020). As a result, it provides preliminary results for the small part of miss-generated roofs (of the order of a few percent of the dataset). Everything is stored in the CityJSON file to render and provide building information.

Besides using airborne point cloud, hybrid models involving terrestrial clouds are used to render the interior of buildings and thus make a link between the neighbourhood and the property unit. The main advantage is that these elements are not only topologically but also semantically linked. No external mapping

is thus necessary to communicate information between the two levels-of-definition.

Figure 3 illustrates a CityJSON *Building*, or at least the abstraction of a *Building*, in which a *BuildingPart* is rendered. The *BuildingPart* is a LAS point cloud loaded remotely from a web server into a MultiPoint geometry. It is a good example of a non-distorted representation thanks to point clouds. Note that, CityJSON can be improved and extended in order to support a deeper level of object definition. It could define object classes such as walls, floor, openings, etc. This improvement will make a step further to a more effortless merging with IFC and thus follows the initial trend engaged by CityGML 3.0 on a semantic level (Kutzner et al., 2020).

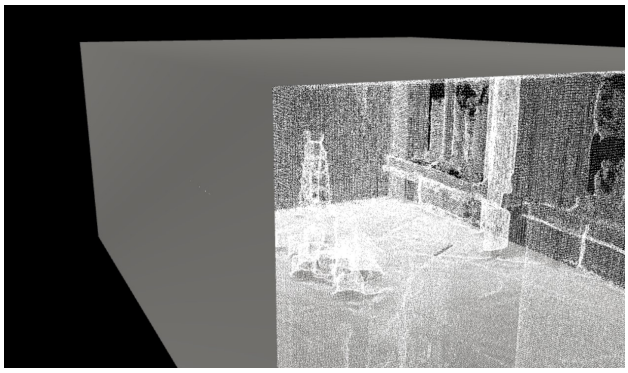

Figure 3. Mixed visualization of a *Building* and its detailed interior as *BuildingPart* (417.138 points)

4.3 Representation of vector fields and dynamic sources

In the third exploratory scenario, point clouds should be used to visualize the distribution of scalar phenomenon as vector fields: climate, airflow and pollution data, etc. It can thus represent information along the three dimensions of space and their entire scale of values. It offers solutions to render simulation results as value-related 3D point cloud (Gautier et al., 2020). Supporting point clouds directly in the schema model is a gain to achieve an integration from a semantic perspective.

About dynamic changes, it is interesting to discuss two points: (1) remote point clouds can change dynamically; (2) point clouds can represent source of dynamic data. Remote data changing dynamically are called Dynamizers in CityGML (Chaturvedi & Kolbe, 2015). In the 3.0 version of CityGML, the dynamizers are now supported natively. Without naming it specifically, external links to dynamic data are alike dynamizers from a conceptual viewpoint. This sets a precedent and will open up CityJSON to further improvements and extension for dynamizers.

Sensors data have a point-like spatialization in location-based applications. Taking the example of a trajectory, it represents many points in an ordered list. Hence, a MultiPoint geometry should provide a solution to represent trajectories in 3D city models. A multiple geometry's discrete management might allow interaction with it also (Liu et al., 2019). Nevertheless, the v1.0.1 CityJSON core is not ready to handle this type of feature natively. It corresponds to no actual type.

Besides, no current city objects type can represent such elements. For instance, this can explain the creation of *FeatureOfInterest* objects, which are objects that are evaluated by the procedure, in the sensors OGC standards: SensorObservationService and SensorThings API OGC. We believe that opening their representation will open possibilities to managing sensors information in a more suitable manner in 3D city models. This extension should permit such development.

4.4 True-to-life representation for complex elements

Finally, point clouds are used in order to render elements that might be distorted during the modelling. For instance, the generalization of which leads to a loss of reliability such as solitary vegetation can reduce the design usefulness. For the urban built environment, examples are churches, minarets, etc. Since the number of geometries is not limited and the different Geometry Objects of a given *CityObject* do not have be of different LoDs, several object geometries can be defined for every city object. One can then have a similar third level defined in point clouds and meshes (point clouds are obligatory LoD 3.x) or a more detailed one if meshes are more generic than the point cloud.

About the solitary vegetation, it thus allows us to avoid the use of *GeometryInstances* and *Templates* from CityJSON. Such templates distort the true-to-life representation of anisotropic object. Nevertheless, it increases the file size given that information is not simplified in a broad template. The sparse nature of the point cloud seems more faithful to reality than a cone or a generic sphere (see Figure 4 and Figure 5). It is seen as a better modelling of the canopy's transparency and a wind resistance.

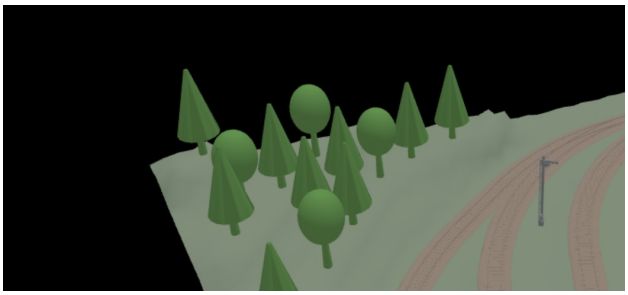

Figure 4. Trees rendering as *GeometryTemplates* from the railway.json dummy

Figure 5. 3D city model in which *SolitaryVegetationObject* are rendered as MultiPoint geometries

An important discussion should take place on the file size and the counterpart taken with the will behind CityJSON. The initial concern of CityJSON is to provide a lightweight alternative to the XML encoding of CityGML. For the reminder, two solutions are proposed to support point clouds in this extension: inline geometries and external links. Both have their pros and cons: while the former does not impose interactions and exchanges of information, the second is the best solution to avoid making a file unwieldy. It is therefore perhaps not a good idea to use inline point clouds in CityJSON. Still, it is now a possibility, but it goes against a wider use of 3D city models

over the web, on small devices, or even if bandwidth is a concern. All the more so as an offline use is impossible.

Quantitative information about file size of the presented example are the following. The example illustrated in Figure 2 shows that the LAS cloud weights 1.35Mb and its CityJSON counterpart is a 1.73Mb. The city model includes 237 objects of which are *Buildings*, *TINRelief* and *SolitaryVegetationObjects*. For the example on Figure 3, the CityJSON file is 1.68Kb large and the remote LAS file is 13.85Mb. The weight ratio is not comparable. Indeed, while the *Building* object counts eight vertices, the MultiPoint for the *BuildingPart* counts a total of 417138 points. Note that the web browser performances are not a concern, as an efficient rendering was not the contribution of this paper. No such problems were encountered during the development of the extended scheme and viewer.

Point clouds are now supported to render and visualize in 3D CityJSON models. Mixed models can be queried and delivered over the web. Future work will study the possibilities of spatial analysis and computation on point clouds in urban built environment.

5. CONCLUSION

This paper proposes a solution for merging point clouds and 3D city models. It consists in an extension of the CityJSON encoding to support point clouds. Two solutions are proposed: inline geometries and external link. It follows the CityGML 3.0 specifications. While the extension is straightforward, its applications are numerous and open possibilities: climate analysis, landscape projects, BIM, etc. Among others, the present research illustrates technical capabilities without going into pure application: object generation fall-back solution, high-detailed features, representation of vector fields and true-to-life representation for complex features. It is mandatory to keep an eye on providing a lightweight encoding: the model size. We therefore encourage users to prefer the external link solution rather than weighing down the models and geometries.

REFERENCES

Biljecki, F., Kumar, K., & Nagel, C. (2018). CityGML Application Domain Extension (ADE): Overview of developments. *Open Geospatial Data, Software and Standards*, *3*(1), 13. https://doi.org/10.1186/s40965-018-0055-6

Brunnhuber, M., May, M., Traxler, C., Hesina, G., Glatzl, R., & Kontrus, H. (2017). Using Different Data Sources for New Findings in Visualization of Highly Detailed Urban Data. *Proceedings of 22nd International Conference on Urban Planning, Regional Development and Information Society*, 637–646.

Buyukdemircioglu, M., & Kocaman, S. (2018). A 3D CAMPUS APPLICATION BASED ON CITY MODELS AND WEBGL. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII–5*, 161–165. https://doi.org/10.5194/isprs-archives-XLII-5-161-2018

Chaturvedi, K., & Kolbe, T. H. (2015). *Dynamizers—Modeling and Implementing Dynamic Properties for Semantic 3D City Models*. https://doi.org/10.2312/udmv.20151348

Discher, S., Richter, R., & Döllner, J. (2019). Concepts and techniques for web-based visualization and processing of massive 3D point clouds with semantics. *Graphical Models*, *104*, 101036. https://doi.org/10.1016/j.gmod.2019.101036

Gautier, J., Christophe, S., & Brédif, M. (2020). VISUALIZING 3D CLIMATE DATA IN URBAN 3D MODELS. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLIII-B4-2020*, 781–789. https://doi.org/10.5194/isprs-archives-XLIII-B4-2020-781-2020

Joshi, M. Y., Selmi, W., Binard, M., Nys, G.-A., & Teller, J. (2020). POTENTIAL FOR URBAN GREENING WITH GREEN ROOFS: A WAY TOWARDS SMART CITIES. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *VI-4/W2-2020*, 87–94. https://doi.org/10.5194/isprs-annals-VI-4-W2-2020-87-2020

Kumar, K., Ledoux, H., & Stoter, J. (2018). Dynamic 3D Visualization of Floods: Case of the Netherlands. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII-4/W10*, 83–87. https://doi.org/10.5194/isprs-archives-XLII-4-W10-83-2018

Kutzner, T., Chaturvedi, K., & Kolbe, T. H. (2020). CityGML 3.0: New Functions Open Up New Applications. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*. https://doi.org/10.1007/s41064-020-00095-z

Ledoux, H., Ohori, K. A., Kumar, K., Dukai, B., Labetski, A., & Vitalis, S. (2019). CityJSON: A compact and easy-to-use encoding of the CityGML data model. *ArXiv:1902.09155 [Cs]*. http://arxiv.org/abs/1902.09155

Liu, D., Peng, J., Wang, Y., Huang, M., He, Q., Yan, Y., Ma, B., Yue, C., & Xie, Y. (2019). Implementation of interactive three-dimensional visualization of air pollutants using WebGL. *Environmental Modelling & Software*, *114*, 188–194. https://doi.org/10.1016/j.envsoft.2019.01.019

Nebiker, S., Bleisch, S., & Christen, M. (2010). Rich point clouds in virtual globes – A new paradigm in city modeling? *Computers, Environment and Urban Systems*, *34*(6), 508–517. https://doi.org/10.1016/j.compenvurbsys.2010.05.002

Nys, G.-A., Billen, R., & Poux, F. (2020). AUTOMATIC 3D BUILDINGS COMPACT RECONSTRUCTION FROM LIDAR POINT CLOUDS. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLIII-B2-2020*, 473–478. https://doi.org/10.5194/isprs-archives-XLIII-B2-2020-473-2020

Nys, G.-A., Poux, F., & Billen, R. (2020). CityJSON Building Generation from Airborne LiDAR 3D Point Clouds. *ISPRS International Journal of Geo-Information*, *9*(9), 521. https://doi.org/10.3390/ijgi9090521

Poux, F., Neuville, R., Van Wersch, L., Nys, G.-A., & Billen, R. (2017). 3D Point Clouds in Archaeology: Advances in Acquisition, Processing and Knowledge Integration Applied to Quasi-Planar Objects. *Geosciences*, *7*(4), 96. https://doi.org/10.3390/geosciences7040096

Spielhofer, R., Fabrikant, S. I., Vollmer, M., Rebsamen, J., Grêt-Regamey, A., & Wissen Hayek, U. (2017). *3D Point Clouds for Representing Landscape Change* [Application/pdf]. https://doi.org/10.3929/ETHZ-B-000171222

Urech, P. R. W., Dissegna, M. A., Girot, C., & Grêt-Regamey, A. (2020). Point cloud modeling as a bridge between landscape design and planning. *Landscape and Urban Planning*, *203*, 103903. https://doi.org/10.1016/j.landurbplan.2020.103903

Virtanen, J.-P., Jaalama, K., Puustinen, T., Julin, A., Hyyppä, J., & Hyyppä, H. (2021). Near Real-Time Semantic View Analysis of 3D City Models in Web Browser. *ISPRS International Journal of Geo-Information*, *10*(3), 138. https://doi.org/10.3390/ijgi10030138

Vitalis, S., Labetski, A., Boersma, F., Dahle, F., Li, X., Arroyo Ohori, K., Ledoux, H., & Stoter, J. (2020). CITYJSON + WEB = NINJA. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *VI-4/W1-2020*, 167–173. https://doi.org/10.5194/isprs-annals-VI-4-W1-2020-167-2020

Vitalis, Stelios, Arroyo Ohori, K., & Stoter, J. (2020). CityJSON in QGIS: Development of an open-source plugin. *Transactions in GIS*, tgis.12657. https://doi.org/10.1111/tgis.12657

Vitalis, Stelios, Ohori, K., & Stoter, J. (2019). Incorporating Topological Representation in 3D City Models. *ISPRS International Journal of Geo-Information*, *8*(8), 347. https://doi.org/10.3390/ijgi8080347

Wang, C., Wen, C., Dai, Y., Yu, S., & Liu, M. (2020). Urban 3D modeling with mobile laser scanning: A review. *Virtual Reality & Intelligent Hardware*, *2*(3), 175–212. https://doi.org/10.1016/j.vrih.2020.05.003